CURSO PRÁTICO 14 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 20,00

INPUT

Vol. 1 N.º 14

NESTE NÚMERO

PROGRAMAÇÃO BASIC

OS COMANDOS PEEK E POKE

Como examinar a memória do computador. Funcionamento dos comandos **PEEK** e **POKE**. Aprenda a utilizar o **POKE** para colocar caracteres na tela. Leitura do cronômetro do Spectrum. Cores no TRS-Color. Como controlar o Apple II..... 261

PROGRAMAÇÃO DE JOGOS

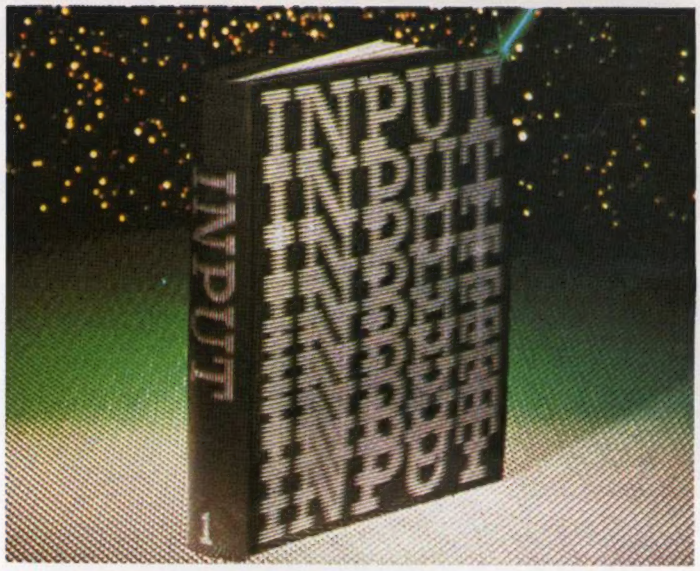
COMO MOVIMENTAR O AVENTUREIRO

Apresentação e descrição dos locais. Como saber onde o jogador se encontra, após cada movimento. Apresentação das direções possíveis. Validação da resposta do jogador. A movimentação do aventureiro 270

APLICAÇÕES

DATILOGRAFIA: ALFABETO COMPLETO

Como acrescentar ao programa as teclas das fileiras superior e inferior. Posição dos dedos nas teclas de outras fileiras. Como mudar as palavras para treinamento. Pratique com todo o alfabeto 276



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: **1. Pessoalmente** — por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em São Paulo os endereços são: Rua Brigadeiro Tobias, 773 (Centro); Av. Industrial, 117 (Santo André); e, no Rio de Janeiro: Rua da Passagem, 93 (Botafogo). **2. Por carta** — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidor Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132 (Jardim Tereza) — CEP 06000 — Osasco — São Paulo. **3. Por telex** — Utilize o nº (011) 33670 ABSA. Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Ltd. — Qta. Pau Varais, Azinhaga de Fetais — 2685, Camarate — Lisboa; Tel. 257-2542 — Apartado 57 — Telex 43 069 JARLIS P.
Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na Agência do Correio. **Atenção**: Após seis meses do encerramento da coleção, os pedidos serão atendidos, dependendo da disponibilidade de estoque. **Obs.:** Quando pedir livros, mencione sempre o título e/ou o autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

NOVA CULTURAL

Editor

VICTOR CIVITA

REDAÇÃO

Diretora Editorial: Iara Rodrigues

Editor chefe: Paulo de Almeida
Editor de texto: Cláudio A.V. Cavalcanti
Editor de Arte: Eduardo Barreto
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Ailton Oliveira Lopes, Dilvacy M. Santos, José Maria de Oliveira, Grace A. Arruda, Monica Lenardon Corradi
Secretária de Redação/Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz
Secretário Gráfico: Antonio José Filho

COLABORADORES

Consultor Editorial Responsável: Dr. Renato M.E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda. Campinas, SP.
Tradução: Maria Fernanda Sabbatini
Adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Raul Neder Porrelli
Coordenação geral: Rejane Felizatti Sabbatini
Editora de Texto: Ana Lúcia B. de Lucena
Assistente de Arte: Dagmar Bastos Sampaio

COMERCIAL

Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Ferrucio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO

Gerente de Produção: João Stungis
Coordenador de Impressão: Atílio Roberto Bonon
Preparador de Texto/Coordenador: Eliel Silveira Cunha
Preparadores de Texto: Ana Maria Dilguerian, Antonio Francelino de Oliveira, Karina Ap. V. Grechi, Levon Yacubian, Maria Teresa Galluzzi, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria do Carmo Leme Monteiro, Maria Luiza Simões, Maria Teresa Martins Lopes.

© Marshall Cavendish Limited, 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.

Edição organizada pela Editora Nova Cultural Ltda.
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Dívisão Gráfica da Editora Abril S.A.

OS COMANDOS PEEK E POKE

- O EXAME DA MEMÓRIA
- ESCREVA NA TELA COM O POKE
- CARACTERES NO SPECTRUM
- CORES NO TRS-COLOR
- COMO CONTROLAR O APPLE II

Os comandos PEEK e POKE permitem examinar e alterar valores diretamente na memória do computador. E você pode usá-los sem dificuldade, incorporados a um programa BASIC.

Muitas vezes, usamos o computador sem nenhuma necessidade de saber como sua memória funciona. Quando escrevemos, por exemplo, **LET A = 67**, o computador por si só seleciona posições de memória livres, chama estas posições de "A" e nelas armazena o valor **67**. Se digitarmos, em seguida, **PRINT A**, o computador saberá exatamente onde encontrar o valor pedido — o processo é automático. Só é necessário indicar ao computador as posições de memória que devem ser usadas quando programamos em código de máquina.

Mas existe em BASIC uma maneira de examinar a memória do computador, para localizar os valores ali armazenados. É possível, também, armazenar valores apropriados na memória, a fim de alterar o comportamento da máquina.

As ferramentas BASIC para isso são o **PEEK** e o **POKE**. **PEEK** permite que se examine o valor armazenado na memória e **POKE** possibilita o armazenamento de um determinado valor.

COMO FUNCIONAM O PEEK E O POKE

Os computadores TRS-80, TRS-Color, MSX, bem como os ZX-81 e Spectrum de 48K, possuem 65 536 posições de memória endereçáveis; portanto, 64K. O Apple e o TK-2000, em suas diversas versões, apresentam diferentes tamanhos de memória. Uma parte dessa memória é a ROM, ou seja, a Memória Apenas para Leitura. Os conteúdos da ROM são fixos; assim, embora possamos examiná-la usando **PEEK**, não podemos utilizar o **POKE** para armazenar novos valores ou alterar os que nela já existem. O restante da memória é a RAM — a Memória de Acesso Aleató-

rio, ou Memória para Leitura e Escrita, como também é conhecida —, na qual se armazenam as variáveis e os programas BASIC. Podemos usar **PEEK** para examinar e **POKE** para armazenar os valores ali contidos.

Este programa permite que você examine qualquer posição de memória ROM ou RAM do computador:

```
10 INPUT "ENDERECO...";A
20 LET N=PEEK(A)
30 PRINT "CONTEUDO...";N
40 GOTO 10
```

Entre com o número que desejar, de 0 a 65 535, e você verá o que existe na posição indicada. Pode ocorrer, porém, que ao usar **PEEK** para examinar determinadas áreas da memória, você receba como resultado um valor fictício e não o valor real lá existente. Os conteúdos da RAM dependerão sempre do que você estiver fazendo com o computador; apenas os conteúdos da ROM são fixos.

Repare que os números são sempre valores inteiros entre 0 e 255 e, em hexadecimal, de 0 a FF. Isso significa que todos constituem bytes simples (um byte contém um número hexadecimal de dois dígitos). Cada posição de memória contém um byte, não sendo possível guardar qualquer número maior em uma posição de memória. Se você somar 1 a FF, em hexadecimal, obterá 0100. Como este resultado inclui dois bytes, será preciso armazená-lo em duas posições de memória: 01 em uma posição e 00 em outra.

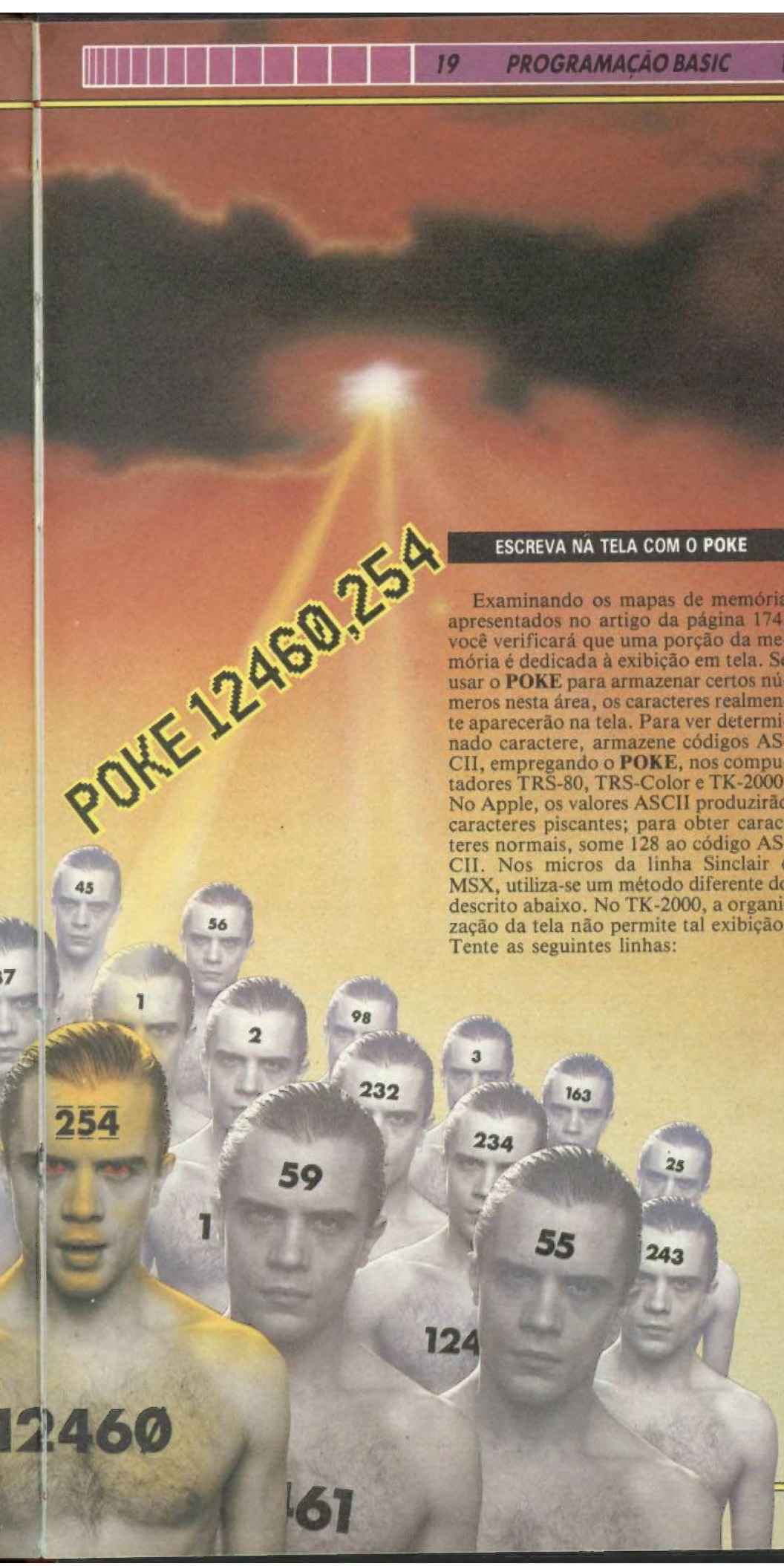
O programa que apresentamos a seguir permite que se empregue o comando **POKE** para guardar números na memória do computador. Neste estágio, armazene apenas bytes simples, ou seja, qualquer número entre 0 e 255.

```
10 PRINT"CONTEUDO...";PEEK(3000
)
20 INPUT"NUMERO....";N
30 POKE 30000,N
40 PRINT"NOVO CONTEUDO";PEEK(30
000)
45 PRINT
50 GOTO 20
```

O programa apresenta primeiro o conteúdo da posição de memória e, em seguida, usa o **POKE** para armazenar ali o número indicado. Depois, mostra uma vez mais o conteúdo, para provar que o número realmente foi armazenado.

Podemos trocar a posição de memória para qualquer outra posição que desejarmos. Repare que nada acontece se tentarmos usar **POKE** na ROM e não causaremos nenhum dano ao sistema se o fizermos. Utilizado em certas áreas da RAM, porém, o **POKE** pode provocar algum desarranjo, mas nada fatal — simplesmente desligue a máquina por um momento, para reiniciar a memória.

Tente agora utilizar o **POKE** para armazenar um número maior do que 255 e veja o que acontece. Nos computadores TRS-Color e Spectrum, você obterá uma mensagem de erro, já que só há lugar para um byte em cada posição de memória.



ESCREVA NA TELA COM O POKE

Examinando os mapas de memória apresentados no artigo da página 174, você verificará que uma porção da memória é dedicada à exibição em tela. Se usar o **POKE** para armazenar certos números nesta área, os caracteres realmente aparecerão na tela. Para ver determinado caractere, armazene códigos ASCII, empregando o **POKE**, nos computadores TRS-80, TRS-Color e TK-2000. No Apple, os valores ASCII produzirão caracteres piscantes; para obter caracteres normais, some 128 ao código ASCII. Nos micros da linha Sinclair e MSX, utiliza-se um método diferente do descrito abaixo. No TK-2000, a organização da tela não permite tal exibição. Tente as seguintes linhas:

TABELA DE CÓDIGO ASCII

Cada caractere que o computador utiliza possui um código numérico, e muitos modelos adotam um código padrão denominado Código ASCII — iniciais de *American Standard Code for Information Interchange* (Código Padrão Americano para Intercâmbio de Informação).

Estes códigos são os números usados em **CHR$** e **ASC** (ou **CODE** no Spectrum). **CHR$** converte o número em um caractere, enquanto **ASC** ou **CODE** faz o inverso, convertendo um caractere em seu código numérico.

Assim, quando o computador guarda uma palavra, o que armazena é o código ASCII do caractere.

Tabela ASCII

Número de código	Caractere ASCII	Número de código	Caractere ASCII
32	space	62	>
33	!	63	?
34	"	64	@
35	#	65	A
36	$	66	B
37	%	67	C
38	&	68	D
39	'	69	E
40	(	70	F
41	)	71	G
42	★	72	H
43	+	73	I
44	,	74	J
45	—	75	K
46	•	76	L
47	/	77	M
48	0	78	N
49	1	79	O
50	2	80	P
51	3	81	Q
52	4	82	R
53	5	83	S
54	6	84	T
55	7	85	U
56	8	86	V
57	9	87	W
58	:	88	X
59	;	89	Y
60	<	90	Z
61	=		

```
POKE 15360,65
```

```
POKE 1024,65
```

```
HOME:POKE 1024,193
```

A letra A deverá aparecer no canto superior esquerdo da tela.

No TRS-Color, se a tela tiver rolado antes de uma tentativa de armazenar caracteres pelo **POKE**, eles não aparecerão em lugar nenhum. Assim, primeiramente limpe a tela, digitando **CLS** e, depois, **<RETURN>** ou **<ENTER>**.

Você deve estar se perguntando qual a vantagem de usar o **POKE** para colocar caracteres na tela, já que dispomos de comandos tão eficientes como o **PRINT A**, **PRINT TAB** ou **PRINT@**. De fato, o **PRINT** é normalmente o método mais fácil e conveniente para posicionar os caracteres. Mas há casos em que o emprego do **POKE** se mostra vantajoso. Por exemplo, se usarmos o **PRINT** para escrever na última posição da tela, esta sempre rolará; com o **POKE**, ao contrário, não haverá nenhum problema. Você pode imaginar o quanto esta característica é útil, sobretudo em jogos, quando queremos mover um caractere por toda a tela.

OS CARACTERES NO SPECTRUM

No Spectrum, as coisas são um pouco diferentes, pois não se pode usar o **POKE** para armazenar caracteres sobre a tela. Um caractere é feito de pontos de uma matriz de oito por oito. E cada linha deve ser armazenada pelo **POKE** separadamente, até se completar o caractere.

Felizmente, as formas dos caracteres são armazenadas na **ROM**; assim, pode-se usar o **PEEK** nesta porção da memória, a fim de examinar cada linha e, então, exibi-la na tela com o **POKE**.

O programa que apresentamos a seguir utiliza o comando **POKE** para exibir a letra A no canto superior esquerdo da tela.

```
10 LET dest=16384
20 FOR a=15880 TO 15887
30 POKE dest,PEEK a
40 LET dest=dest+256
50 NEXT a
```

A linha 10 define o endereço da primeira linha dos caracteres da tela. O **FOR...NEXT** permite o acesso à ROM, onde o caractere está armazenado e a linha 30 imprime a primeira linha deste caractere na tela. A linha 40 incrementa o endereço da tela em 256, dando acesso à próxima linha, logo abaixo da primeira.

O método funciona, mas, por ser um tanto lento, quase sempre é mais vantajoso usar o **PRINT AT** ou **PRINT TAB** no Spectrum.

O USO DO PEEK NA ROM E NA RAM

Quando executamos o primeiro programa que examina a memória do computador, obtivemos apenas números entre 0 e 255. Mas muitos desses números são, na realidade, códigos de letras em ASCII, alguns dos quais formam palavras ou mesmo sentenças completas. Se você não conhece os códigos ASCII, veja a tabela da página 263.

O programa seguinte lê toda a memória ROM do computador e converte os números em caracteres, antes de imprimi-los na tela:

```
10 FOR A=0 TO 16383
20 LET N=PEEK A
30 IF N>31 AND N<127 THEN PRINT
 CHR$N;
40 NEXT A
```

```
10 FOR A=&H8000 TO &HBFF
20 N=PEEK(A)
30 IF N>31 AND N<127 THEN PRINT
 CHR$(N);
40 NEXT A
```

```
10 FOR A=&H0 TO &H7FFE
20 N=PEEK(A)
30 IF N>31 AND N<127 THEN PRINT
 CHR$(N);
40 NEXT A
```

```
10  FOR A = 53248 TO 65535
20 N =  PEEK (A)
30  IF N > 31 AND N < 127 THEN
 PRINT  CHR$ (N);
40  NEXT A
```

As linhas 10 e 40 determinam a repetição do processo em toda a memória; a linha 20 examina cada posição, usando o **POKE**, e a linha 30 converte o número em um caractere e o imprime. Esta linha ainda limita o intervalo de números que serão convertidos, evitando que o computador imprima códigos de controle ou símbolos gráficos.

Para o TRS-80, use o programa do TRS-Color com a linha 10 do programa do Spectrum.

Pode-se também imprimir o conteúdo da RAM do mesmo modo, bastando trocar os endereços de memória da linha 10. Os endereços dados abaixo imprimem parte da área em que os programas BASIC estão armazenados, permitindo que você veja como o seu próprio programa está guardado na memória. Convém desligar o computador por um segundo antes de rodar este programa, para evitar que a RAM permaneça carregada com restos de outros programas. Aqui está a nova linha 10:

```
10 FOR A=23755 TO 65000
```

```
10 FOR A=16510 TO 30000
```

```
10 FOR A=&H1E00 TO &H1F00
```

Use **FOR 17385 TO 32767** para sistemas de 16K e os finais **49151** para 32K e **65535** para 48K.

```
10  FOR A = 2048 TO 32767
```

```
10 FOR A=&H8000 TO &HF37F
```

Experimente examinar outras áreas da RAM da mesma maneira.

APLICAÇÕES

Até aqui examinamos a memória do computador e usamos **POKE** para exibir caracteres na tela. Isso nos deu uma boa noção de como o **PEEK** e o **POKE** funcionam, mas não do quanto são úteis.

Para as aplicações mais importantes, precisaremos examinar posições específicas de memória. No Spectrum, a posição de memória 23609 controla o som que o teclado faz quando se pressiona uma tecla. Normalmente, ela contém 0, que provoca um pequeno clic; mas, com um número maior — que pode chegar a 255 —, obtém-se um longo bip. **POKE 23609,80** faz um ruído razoável.

As possibilidades de uso do **PEEK** e do **POKE** dependem principalmente do computador que você tem.

Retomando o exemplo anterior: os teclados dos demais computadores não fazem ruído quando uma tecla é pressionada; não têm posição de memória que controle o som do teclado. Assim, um efeito que requer cinco ou seis **PEEK** ou **POKE** em determinado computador pode ser obtido com uma palavra-chave BASIC em outra máquina.

O Apple faz um bom uso do **PEEK** e do **POKE**. O Spectrum, o TRS-80 e o TRS-Color utilizam-no ocasionalmente. No MSX o uso destes comandos é mais raro.

Aqui estão algumas aplicações para o seu computador.

S

Já vimos um **POKE** modificar o som do teclado. O **POKE** seguinte altera o espaço de tempo que uma tecla leva para iniciar a auto-repetição — o que é útil em jogos de velocidades, nos quais uma tecla precisa ser pressionada para mover um caractere. Neste caso e nos seguintes, **X** é uma variável que devemos ajustar por meio do comando.

```
POKE 23561,X
```

Normalmente, quando se liga o computador, **X** é ajustado para 35. Assim, use um número menor que 35 para diminuir o intervalo de espera e um número maior para obter um intervalo mais longo.

Pode-se também alterar a repetição automática com:

```
POKE 23562,X
```

Este tempo **X** é, em geral, igual a 5. Assim, use **X = 1** para uma auto-repetição rápida e **X = 25** para uma bem lenta.

CONTROLE DE TEMPO

O Spectrum não tem acesso ao cronômetro interno via teclado. Para utilizá-lo, precisaremos usar o **PEEK** na memória. O tempo é armazenado como três bytes em três posições consecutivas de memória.

```
PRINT(PEEK 23672+256*PEEK 23673
+65536*PEEK 23674)
```

... informará quantos cinqüenta avos de segundo o seu Spectrum está ligado desde o último **NEW**. Pressione **NEW** para reiniciar o cronômetro ou **POKE 0** em cada uma das três posições.

MAIS POKE

Apresentaremos aqui mais duas aplicações do **POKE**, que serão úteis quando você quiser escrever programas para outra pessoa. A primeira delas assegura que todas as letras digitadas apareçam em tipo maiúsculo.

```
POKE 23568,8
```

Esta instrução é útil quando se pretende que todos os dados de entrada sejam consistentes. Use **POKE 23658,0** para retornar ao normal.

Uma outra possibilidade é a alteração do cursor para qualquer caractere do teclado — ou mesmo qualquer palavra-chave, dependendo do número que foi usado no **POKE**:

```
10 FOR X=1 TO 255
20 POKE 23617,X
30 INPUT a$
40 NEXT X
```

A linha 30 faz um **INPUT a$** apenas para que o cursor apareça na tela. Tão logo entre algum dado via **INPUT**, o programa mostrará o cursor seguinte. Os mais interessantes são aqueles por volta de **X** = 200 até 230. Por exemplo, **X = 210** exibe **<CONTINUE>** como um cursor, e **X = 225** exibe um sinal de interrogação.

T

O **PEEK** e o **POKE** são também utilizados para a auto-repetição. O TRS-Color normalmente não tem teclado auto-repetitivo. Para mover objetos em jogos, por exemplo, é preciso manter a tecla pressionada, soltá-la muito rapidamente, voltar a pressioná-la e, assim, sucessivamente. Uma maneira de evitar esse cansativo procedimento é usar **PEEK** para verificar que tecla está sendo pressionada, o que nos permite deixar o dedo sobre a tecla pelo tempo que quisermos.

Esse método já foi empregado na seção de Programação de Jogos. Veja como funciona: quando pressionamos uma tecla, um código numérico especial é colocado em uma das seis posições de memória. O **PEEK** checa, então, estas posições, para verificar que tecla está sendo pressionada. O emprego deste recurso é muito simples. O programa seguinte, por exemplo, usa teclas de cursor para desenhar na tela:

```
10 PMODE 0,1
20 PCLS
30 SCREEN 1,1
40 X=127:Y=95
50 IF PEEK(341)=223 THEN Y=Y-2
60 IF PEEK(342)=223 THEN Y=Y+2
70 IF PEEK(343)=223 THEN X=X-2
80 IF PEEK(344)=223 THEN X=X+2
90 IF X<0 OR X>255 THEN X=-255*
(X<0)
100 IF Y<0 OR Y>191 THEN Y=-191
*(Y<0)
110 PSET (X,Y,5)
120 GOTO 50
```

A tabela da página 265 mostra o código numérico gerado para cada tecla e a posição de memória em que elas são armazenadas. Se, por exemplo, pressionássemos A, **PEEK (339)** seria igual a 267. Podemos verificar, assim, de onde vieram os números escritos após os **PEEK** nesse último programa.

TELA MULTICOLORIDA

O programa seguinte traça linhas de diferentes cores na tela. Colocando-as perto umas das outras, obteremos novos matizes. Você pode pressionar qualquer tecla durante a execução do programa, se quiser interromper o desenho e examinar o número para determinado matiz.

```
10 PMODE 3,1:SCREEN 1,0:PCLS3
20 FOR K=0 TO 255
30 POKE 178,K
40 LINE(78,46)-(178,146),PSET,B
F
50 IF INKEY$<>"" THEN 80
60 NEXT
70 GOTO 70
80 CLS:PRINT K
```

A posição de memória 178 controla a cor do primeiro plano. Esta varia, em geral, de 0 a 3 (4 cores), mas, se usarmos o **POKE** para armazenar um valor maior que **K = 3**, obteremos novas listras coloridas.

Utilizamos **PSET** na linha 40 porque estamos manipulando cores de primeiro plano. Para trabalhar as cores de plano de fundo, troque **PSET** por **PRESET** e substitua o valor 178 na linha 30 por 179.

Parando o programa, poderemos examinar o valor de **K** para determinado matiz e, assim, usar esta cor em outros programas gráficos.

COMO ALTERAR A VELOCIDADE

A possibilidade de acelerar o computador é interessante, sobretudo, para jogos — você pode fazê-los andar mais depressa se, por exemplo, o jogador atingir um determinado número de pontos.

Utilize o primeiro **POKE** para acelerar e o segundo para voltar ao normal.

```
POKE 65495,1
POKE 65494,1
```

Pode-se usar qualquer número em lugar do 1, pois o efeito será o mesmo.

Existe apenas um problema com estes **POKE** — como eles não funcionam em qualquer processador, corre-se o risco de que o programa prejudique o sistema. Mas não custa nada experimentá-los uma vez em seu computador.

E, agora, se você considera determinado jogo muito rápido, aqui está uma maneira de desacelerar a saída na tela.

```
POKE 359,60
```

Para voltar ao normal, use **POKE 359,67**, mas não no CP-400 Color, pois ele desativa o retorno automático ao modo texto.

No Apple é possível controlar a porção da tela que iremos utilizar. O endereço 32 contém a margem esquerda; 33, a largura máxima da linha; 34, o número de linhas da margem superior e 35, da margem inferior. Modificando os valores desses endereços com **POKE**, pode-se programar a janela de textos. É o que faz o programa seguinte.

```
5  HOME : VTAB 5: HTAB 11
10  POKE 32,10: POKE 33,20
20  POKE 34,4: POKE 35,20
30  PRINT "JANELA ";
40  GOTO 30
```

Outra utilidade destes comandos é possibilitar uma "varredura do teclado", verificando o valor contido na posição −16384 — sendo maior que 127, sabe-se que alguma tecla foi pressionada. Na realidade, esta posição de memória contém o código ASCII do caractere da tecla pressionada mais 128.

Sempre que fizermos esse tipo de "varredura", devemos colocar o valor 0 na posição −16368, para que uma nova verificação seja possível. O desrespeito a esta regra pode prejudicar seus programas. O programa que se segue exemplifica a utilização desse recurso.

```
10  HOME
20 A =  PEEK ( - 16384): POKE
- 16368,0
30  IF A > 127 THEN  PRINT  CHR
$ (A - 128);
40  GOTO 20
```

O programa funciona como uma máquina de escrever. Os caracteres correspondentes às teclas pressionadas são mostrados no vídeo.

O uso do **PEEK** também possibilita a produção de sons. Toda vez que verificamos o conteúdo do endereço −16336, um clic é produzido. Faça uma experiência, adicionando as linhas abaixo ao programa anterior.

```
30  IF A > 127 THEN  PRINT  CHR
$ (A - 128);: FOR I = 1 TO 10:X
 =  PEEK ( - 16336): NEXT
```

COMO ALTERAR A TELA

O Apple pode mostrar texto e gráficos de baixa e alta resolução em duas "páginas diferentes". Alguns **POKE** são capazes de modificar rapidamente as características da tela.

```
POKE -16304,0
```

Troca a tela de textos pela tela gráfica, sem limpar esta última.

```
POKE -16303,0
```

Troca a tela gráfica de qualquer tipo pela tela de textos, sem redefinição da janela de textos.

```
POKE -16302,0
```

Permite a exibição de gráficos em tela cheia, ou seja, impede o aparecimento de texto nas quatro linhas inferiores da tela gráfica.

```
POKE -16301,0
```

Troca a tela cheia por gráfico mais texto, ou seja, faz reaparecer as quatro linhas de texto na parte inferior da tela.

```
POKE -16300,0
```

Passa da página 2 para a página 1. Não limpa a tela nem move o cursor.

```
POKE -16299,0
```

Passa da página 1 para a página 2, da mesma maneira.

```
POKE -16298,0
```

Passa do modo gráfico de alta resolução para o modo texto, sem limpar a tela.

```
POKE -16297,0
```

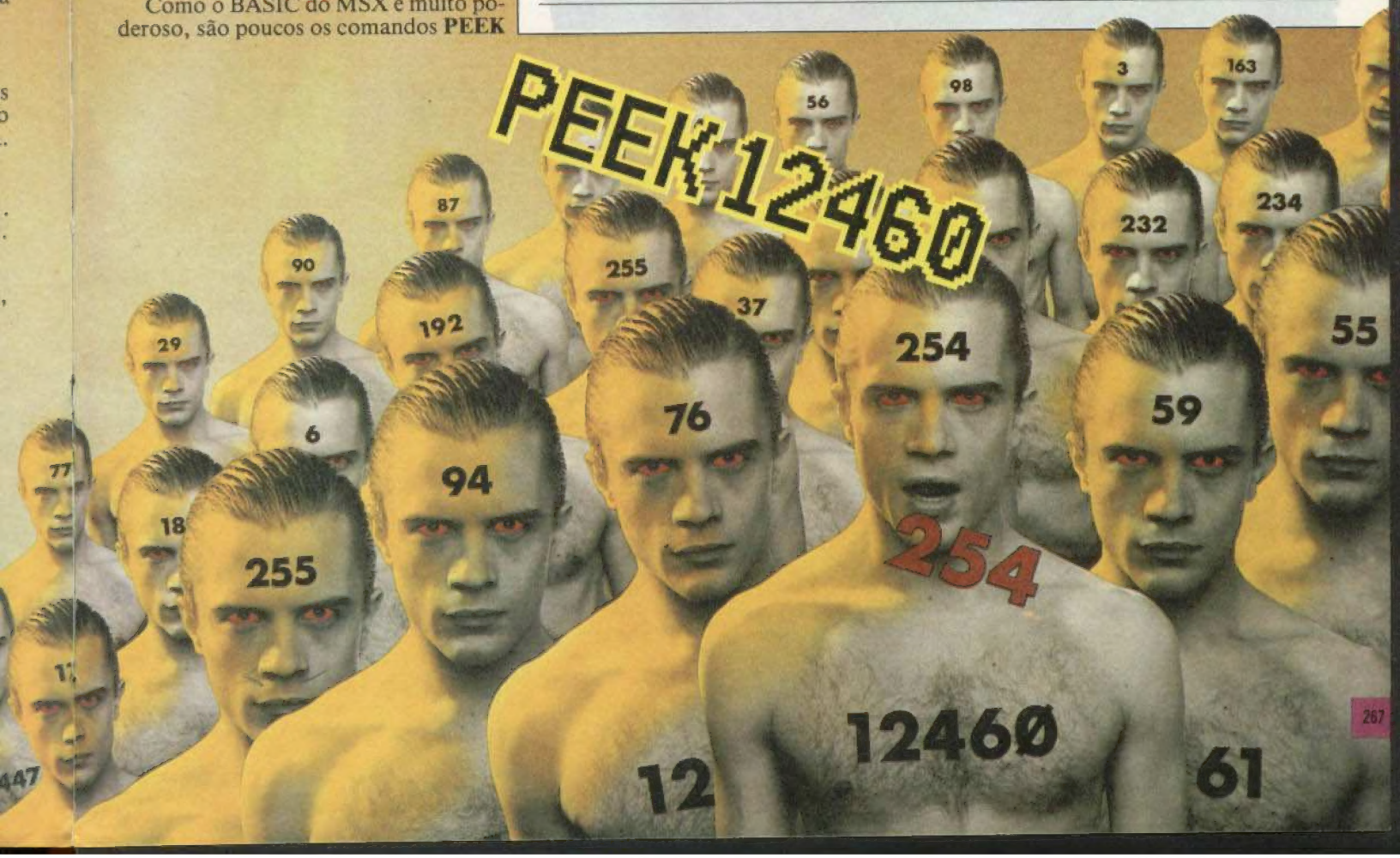
Passa do modo texto para o modo gráfico de alta resolução.

Como o BASIC do MSX é muito poderoso, são poucos os comandos **PEEK**

PEEK e POKE de teclado do TRS-Color

Endereço / Número de código

	191	223	239	247	251	253	254
338	ENTER	8	Ø	X	P	H	@
339	CLEAR	9	1	Y	Q	I	A
340		:	2	Z	R	J	B
341		;	3	↑	S	K	C
342		,	4	↓	T	L	D
343		—	5	←	U	M	E
344		•	6	→	V	N	F
345		/	7	SPACE	W	O	G



e **POKE** realmente úteis.

O modo como este computador organiza sua memória de vídeo impede que coloquemos caracteres diretamente na tela, usando **POKE**. Para isto, existe no BASIC do MSX o comando **VPOKE**, bem como o correspondente **VPEEK**. Os 16 kbytes da memória de vídeo só podem ser acessados por meio desses dois comandos.

O computador utiliza essa memória não só para mostrar caracteres e gráficos; parte dela armazena informações para o vídeo, tal como o padrão das letras do conjunto de caracteres.

O modo como isso é feito — ou seja, a organização da memória de imagem — varia conforme o tipo de tela escolhida pelo comando **SCREEN**.

Para que não tenhamos que decorar uma grande quantidade de endereços de memória, o MSX usa variáveis do sistema para guardar os endereços.

Vamos tentar entender esta organização. No modo texto de 40 colunas (**SCREEN 0**), a variável **BASE (0)** contém o endereço inicial de uma porção da memória denominada tabela de nomes. Qualquer valor colocado em uma das 960 posições, a partir deste endereço, fará surgir na tela o caractere de código ASCII correspondente. Assim, o programa seguinte enche a tela com o caractere A, cujo código é 65.

```
10 SCREEN 0
20 FOR I=0 TO 959
30 VPOKE BASE(0)+I,65
40 NEXT
```

Note que, para colocarmos o caractere A na i-ésima posição da tela, devemos usar **VPOKE BASE (0) + I,65**.

No modo texto de 32 colunas, **BASE (5)** contém a tabela de nomes e existem 768 posições na tela. Assim, temos um programa análogo.

```
10 SCREEN 1
20 FOR I=0 TO 767
30 VPOKE BASE(5)+I,65
40 NEXT
```

O modo gráfico de alta resolução também possui uma tabela de nomes, cujo endereço inicial fica em **BASE (10)**, com 768 posições. O computador multiplica por 8 o número ali colocado por **VPOKE**. Em seguida usa o resultado dessa operação para obter o padrão e as cores daquela posição. O padrão dos pontos é armazenado em outra parte da memória de vídeo, cujo endereço inicial está em **BASE (12)**. Outra posição de memória é reservada para as cores dos pontos e seu início está em **BASE (11)**. Para entender melhor o processo, digite o seguinte programa:

```
10 SCREEN 2
20 FOR I=0 TO 767
30 VPOKE BASE(10)+I,0
40 NEXT
50 FOR I=0 TO 7
60 VPOKE BASE(11)+I,RND(1)*256
70 VPOKE BASE(12)+I,RND(1)*256
80 NEXT
90 GOTO 50
```

As linhas de 20 a 40 enchem a tabela de nomes com o número 0. Este número é multiplicado por 8, dando 0. Isso significa que o computador irá aos oito primeiros endereços a partir da posição 0 na tabela de padrões, para obter o padrão; e aos oito primeiros endereços a partir da posição 0 da tabela de cores, para obter a combinação de cores destas posições. Como ainda não há nada nas duas tabelas, nenhum desenho é feito a princípio.

As linhas de 50 a 80 preenchem as oito primeiras posições, a partir do endereço 0, da tabela de padrões com números aleatórios. Fazem o mesmo com os endereços da tabela de cores. Estes são justamente os locais onde o computador vai buscar a cor e o padrão de posições da tabela de nomes que tenham o número 0. Como a tabela de nomes está cheia de zeros, todas as posições do terço superior da tela assumirão a mesma cor e padrão simultaneamente. A linha 90 repete o processo.

Você deve estar se perguntando por que o efeito só foi obtido no terço superior da tela. A explicação está no fato de que 255 é o maior valor que um endereço pode conter. Assim, o computador procurará o padrão e a cor de um determinado endereço da tabela de nomes da seguinte forma: o terço superior da tela tem suas cores e padrões nos endereços de 0 a 2055 das tabelas de cores e padrões; o terço médio, dos endereços de 2048 a 4096; e o terço inferior, de 4097 a 6143. Mude as seguintes linhas no programa anterior, para ver o mesmo efeito no terço médio da tela:

```
60 VPOKE BASE(11)+I+2048,RND(1)
*256
70 VPOKE BASE(12)+I+2048,RND(1)
*256
```

No modo gráfico de baixa resolução, a tabela de nomes tem endereço inicial guardado em **BASE (15)**. O número colocado em uma das 768 posições desta tabela é multiplicado por 8, para que obtenhamos a posição da tabela de cores em que estão armazenadas as cores da posição. A tabela de cores tem endereço inicial em **BASE (17)**. Não há tabela de padrões. Mude as seguintes linhas no programa anterior, para compreender melhor o processo:

```
10 SCREEN 3
30 VPOKE BASE(15)+I,0
60 VPOKE BASE(17)+I,RND(1)*256
```

Nas telas de texto, boa parte da memória é utilizada para armazenar os padrões das letras — de uma forma semelhante aos padrões gráficos. Assim, quando colocamos na tabela de nomes um código ASCII, o computador multiplica este valor por 8, para obter a posição em que a forma de letra está armazenada na tabela de padrões. O endereço inicial desta tabela está em **BASE (2)**, na tela de 40 colunas. O programa a seguir mostra na tela estes padrões, depois de lê-los com **VPEEK**.

```
10 SCREEN 0
20 FOR I=65 TO 191
30 VPOKE BASE(0)+1,I
40 FOR J=0 TO 7
50 A=VPEEK(BASE(2)+8*I+J)
55 Z$="00000000"+BIN$(A)
60 LOCATE 10,10+J:PRINT RIGHT$(
Z$,8)
70 NEXT J:FOR K=1 TO 500:NEXT K
80 NEXT I
```

Mude algumas linhas para obter o mesmo resultado na tela de 32 colunas, onde a tabela de padrões fica em **BASE (5)**.

```
10 SCREEN 1
30 VPOKE BASE(5)+1,I
50 A=VPEEK(BASE(7)+8*I+J)
```

Na tela de 32 colunas podemos obter também caracteres coloridos. Para isso, existe uma tabela de cores com apenas 32 posições. O número reduzido de posições significa que cada conjunto de oito caracteres terá necessariamente a mesma cor, conforme seu código ASCII — não conforme sua posição na tela.

O programa seguinte demonstra isto, imprimindo todo o conjunto de caracteres e depois enchendo a tabela de cores — **BASE (6)** — com cores aleatórias.

```
10 SCREEN 1
20 FOR I=0 TO 255
30 VPOKE BASE(5)+I,I
40 NEXT I
50 FOR I=0 TO 31
60 VPOKE BASE(6)+I,RND(1)*256
70 NEXT
80 GOTO 50
```

NÃO DESANIME

A organização da memória de vídeo do MSX é complicada. Nossa intenção foi motivá-lo a conhecer melhor seu computador; em outra oportunidade, você poderá compreender com mais clareza a organização desta memória.

MAIS CÓDIGOS DE CONTROLE

Os códigos de controle podem ser usados em um programa para acionar funções especiais e substituir comandos em BASIC. Veja seu emprego em micros de diferentes linhas.

Anteriormente, vimos como utilizar códigos de controle dentro de um programa em BASIC, por meio da função **CHR$**. Esta retorna o caractere correspondente a um determinado código numérico ASCII. Diversos exemplos foram dados para micros da linha TRS-80.

Agora vamos examinar como micros de outras linhas podem trabalhar com códigos de controle.

Alguns computadores não permitem o acesso a todos os códigos de controle. Os da linha Apple II, por exemplo, ao contrário dos da linha TRS-80, têm poucos usos para a função **CHR$** com esta finalidade.

Nos micros da linha Apple, como veremos adiante, a entrada dos códigos de controle é feita comumente por meio do próprio teclado, pois este dispõe de uma tecla <CONTROL>, que é pressionada em combinação com outras.

COMANDOS DO DOS

Emprega-se muito a função **CHR$** para acionar comandos do DOS (sistema operacional de disquetes), a partir de programas em BASIC. O código de controle que possibilita o uso de um comando do DOS em um programa é o 4.

Um comando do DOS em geral só pode ser utilizado de forma direta (sem ser precedido de uma linha de programa). Se quisermos, por exemplo, examinar a lista de arquivos em um disquete, digitamos o comando direto:

```
CATALOG
```

Se usarmos dentro de um programa:

```
100 CATALOG
```

... o comando não funcionará quando o programa for executado. Para que isto ocorra, devemos colocar:

```
100 PRINT CHR$(4)+"CATALOG"
```

Quando esta linha é executada, o interpretador BASIC examina o primeiro caractere da mensagem a ser transmitida para o vídeo, com o comando **PRINT**. Se ele tiver o código ASCII igual a 4, a mensagem será enviada ao sistema operacional, como se fosse uma frase de comando.

ECONOMIZE MEMÓRIA

Existem outras maneiras de se colocar o código 4 em uma linha **PRINT**. O seguinte expediente economiza memória, quando muitas linhas **PRINT** com **CHR$ (4)** são necessárias ao programa:

```
90 LET C4$=CHR$(4)
100 PRINT C4$;"CATALOG"
110 PRINT C4$;"RUN PROG2"
```

Pode-se também pressionar as teclas <CONTROL> e D simultaneamente, logo depois das primeiras aspas que se seguem ao comando **PRINT**, e, em seguida, continuar a digitar normalmente o comando do DOS que se deseja.

Este método apresenta um inconveniente: o código de controle 4 (que é inserido por <**CRTL**> <**D**>) fica "invisível" na listagem, podendo não ser notado posteriormente e nem aparecer na listagem da impressora.

Em geral, o uso da tecla <CONTROL> nos micros Apple dá acesso a uma série de funções via teclado.

Existe uma boa razão para se usar códigos de controle nos micros da linha Spectrum (como o TK-90X): economizar memória. Os códigos podem ser empregados para substituir as instruções **PAPER, INK, BRIGHT** e **FLASH**. São digitados ao se pressionar as teclas <CAPS SHIFT> e <SYMBOL SHIFT> e, em seguida, um número (o que equivale à tecla <CONTROL> de outros micros). O número determinará qual instrução está sendo substituída.

Para **PAPER**, entre simplesmente o número da cor (1 a 7). Por exemplo, usamos o 2 para a cor vermelha.

Para **INK**, pressione <CAPS SHIFT> com o número da cor.

Para obter **BRIGHT**, pressione o número 9 e, para desligá-lo, pressione 8.

Finalmente, para obter **FLASH**, pressione <CAPS SHIFT>, seguido do número 9. Para desligá-lo, use o número 8. Lembre-se de pressionar <CAPS SHIFT> e <SYMBOL SHIFT> primeiro, de cada vez. Veja quanta memória foi economizada em uma linha como esta:

```
10 PRINT PAPER 2;INK 6;"MENU'
```

As palavras-chave **PAPER** e **INK** e os dois pontos e vírgulas ocupam 1 byte de memória cada, enquanto os números 2 e 6 ocupam seis. Usando os códigos de controle, a linha ocupará apenas 1 byte para o par de teclas SHIFT, mais 1 byte para o código de controle. Isto significa uma economia de seis bytes por instrução. Assim, um programa que usasse vários comandos de cor ficaria bem menor em tamanho.

Estes comandos precisam ser digitados dentro de aspas para que funcionem corretamente. Portanto, no exemplo anterior, digite primeiro:

```
10 PRINT "
```

... e em seguida digite os códigos de controle para **PAPER 2** e **INK 6**, conforme foi explicado acima, seguidos pelo restante da linha. Não se esqueça das aspas finais.

Os códigos não ocupam espaço visível na listagem, mas funcionam imediatamente, produzindo texto colorido não só na listagem como na tela.

Pode-se utilizar este recurso simplesmente para enfatizar determinados trechos de uma listagem de programa na tela. Neste caso, coloque os códigos de controle fora das aspas, a não ser que você queira que as cores também apareçam na tela.

INCONVENIENTE

O procedimento descrito acima apresenta um inconveniente: os códigos de controle ficam "invisíveis" na tela e na listagem da impressora, não podendo ser notados posteriormente. Se você quiser mudar alguma cor de frente (**INK**) ou de fundo (**PAPER**) no mesmo programa, precisará apagar a parte inicial da linha (ou até ela toda, se não souber localizar onde estão os códigos de controle) e inserir novos códigos pelo método acima.

COMO MOVIMENTAR O AVENTUREIRO

Para descobrir os mais estranhos lugares sem sair de casa, basta entrar no mundo da aventura. Veja aqui como proceder para dar movimento ao aventureiro e iniciar as explorações.

Você já digitou o programa contendo as descrições dos locais que fazem parte desta aventura? Então deve estar ansioso para fazer com que o jogador comece a explorá-los, possibilitando-lhe deslocar-se de um local para outro. Para que isso aconteça, você precisará definir os movimentos possíveis a partir de um determinado local; deverá estabelecer critérios que permitam julgar as respostas dadas pelo jogador e, finalmente, determinar os caminhos que podem ser percorridos.

Apresentaremos, adiante, algumas rotinas necessárias ao desenvolvimento do programa de aventuras. Estas rotinas serão responsáveis pela correta descrição da localização do jogador e pela apresentação das saídas que lhe serão permitidas. Você digitará cada opção e aprenderá a escrever a seção de programa que movimenta o jogador no mundo da aventura, de acordo com a opção que escolheu.

INÍCIO

É fundamental conhecer a localização exata do jogador, a qualquer momento, dentro do jogo. Para isso, o programa usa uma variável **L**, que indica Local. O valor dessa variável muda de acordo com a localização do jogador, a cada movimento realizado.

Para começar, você deve informar ao computador qual a posição do jogador no início da aventura. A primeira seção de um programa para fazer isso é dada em seguida. Carregue a parte do programa que você já possui, por meio do comando **LOAD**, e adicione estas novas linhas:

```
100 CLS : LET DA=0: LET TA=0:
LET LA=0
270 REM **POSICAO INICIAL**
280 LET L=15
290 GOTO 330
```

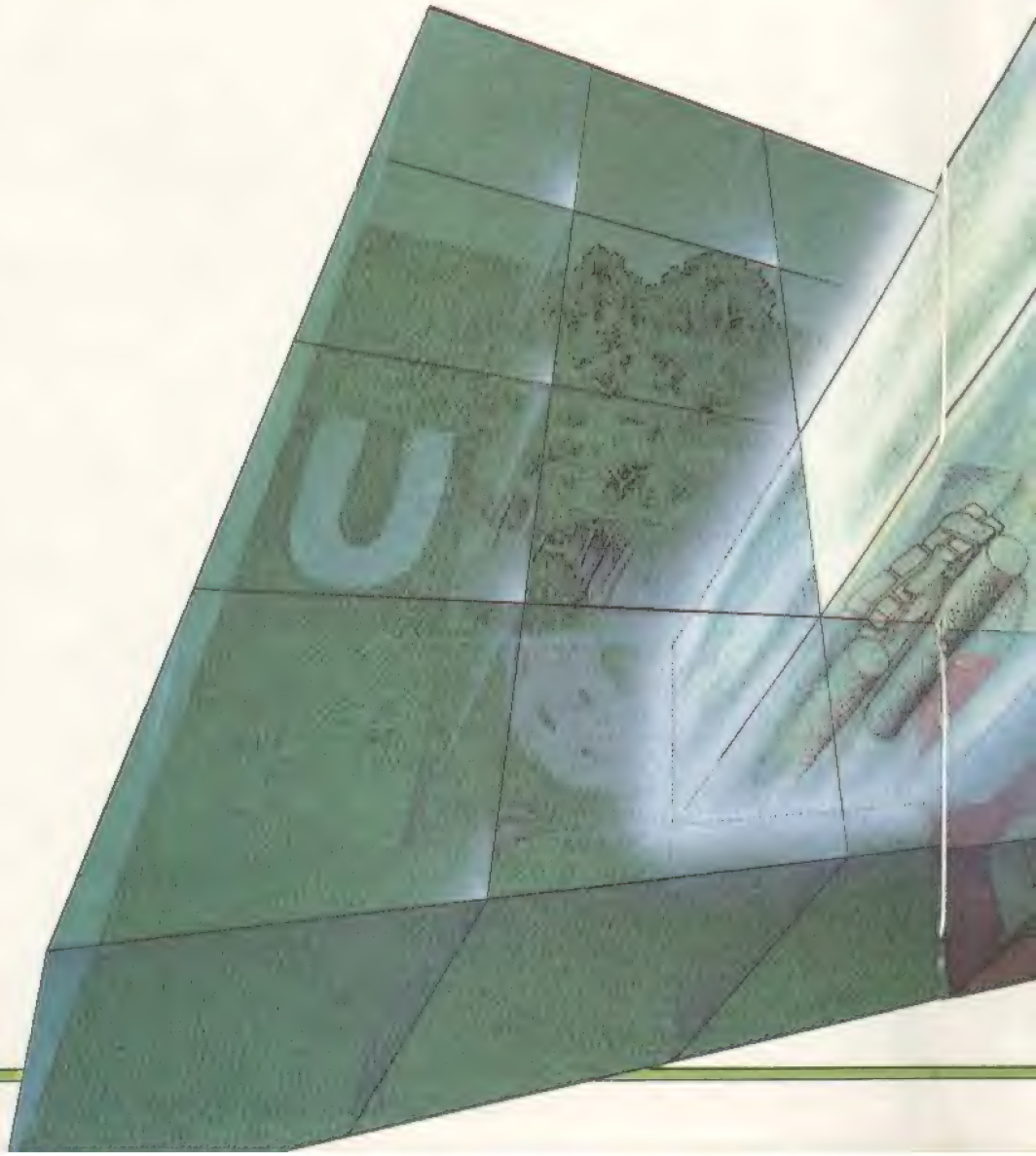
```
270 REM **POSICAO INICIAL**
280 L=15
290 GOTO 330
```

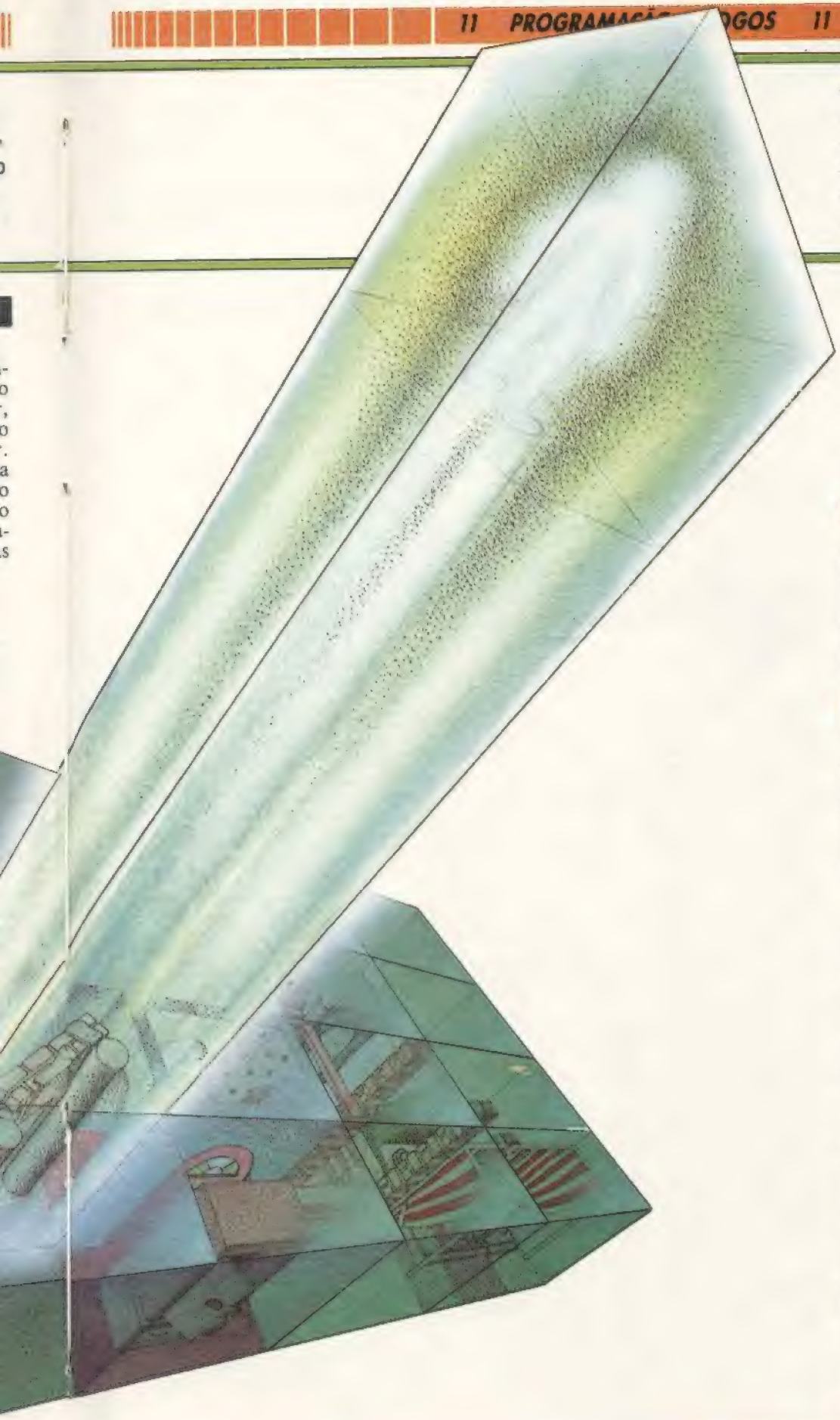
O número 15 indica a localização da porta de entrada da Cidade Oculta. Se você desejar que o jogador inicie a aventura em outro local, bastará mudar o valor da variável **L**. Adiante, você verá como ajustar o valor de **L** durante o jogo, de modo que passe a indicar localização diferente. Por enquanto, devemos preparar o programa para receber do jogador informações indicativas do caminho que pretende seguir.

RESPOSTAS

Para que o computador possa compreender e agir corretamente de acordo com as respostas dadas pelo jogador, você deve dar à máquina um conjunto de palavras que ela passe a identificar.

Neste estágio, o computador precisa conhecer apenas as quatro direções, o que pode ser feito por meio do cordão **R$**. Esse cordão é carregado com as palavras indicativas de direção, colocadas em uma instrução **DATA**.





- APRESENTAÇÃO E DESCRIÇÃO DOS LOCAIS
- DIREÇÕES POSSÍVEIS
- A MOVIMENTAÇÃO DO AVENTUREIRO

S

```
110 REM **PREPARACAO DAS MATRI
ZES DE RESPOSTAS**
120 DIM R$(19,40): DIM R(19)
130 FOR K=1 TO 4: READ R$(K),R
(K): NEXT K
150 DATA "NORTE",1,"SUL",1,"LE
STE",1,"OESTE",1
```

TK

```
110 REM**PREPARAR MATRIZES DAS
RESPOSTAS**
120 DIM R$(19),R(19)
130 FOR K=1 TO 4:READ R$(K),R(K
):NEXT
150 DATA NORTE,1,SUL,1,LESTE,1,
OESTE,1
```

Observe que as matrizes foram dimensionadas na linha 120, podendo agora armazenar todas as respostas que o jogo requer. Como precisamos, inicialmente, de quatro direções, apenas os quatro primeiros elementos das matrizes **R$** e **R** serão utilizados. A instrução **FOR...NEXT**, na linha 130, que lê **R$** e **R** por meio do comando **READ**, varia, assim, de um a quatro. As direções e os valores a ela associados estão colocados na instrução **DATA**, na linha 150.

Essas informações, entretanto, ainda não podem ser usadas pelo jogador. Antes, será necessário dotar o programa de uma rotina que indique onde ele está localizado.

COMO ENCONTRAR O LOCAL

Para que o jogador saiba onde se encontra, após efetuar cada movimento, é necessário fornecer-lhe uma descrição de sua posição. Você já digitou esta des-

crição. Ela está nas linhas de comentários, que se iniciam com a palavra **REM**. Agora, só nos falta uma rotina que associe o número contido na variável **L**, que indica a posição, à descrição correspondente. Neste ponto, os usuários do Spectrum devem digitar uma rotina extra:

```
 20 DIM G(11,4): POKE 23658,8
 30 FOR N=1 TO 4: FOR M=1 TO
 11: READ G(M,N): NEXT M: NEXT
N
 40 DATA 0,0,0,5020,0,0,5050,
 5080,0,5110,0
 50 DATA 5140,0,0,5180,5210,
 5240,5270,5300,0,0,0
 60 DATA 0,5330,0,5360,0,0,0,0
 ,0,0,0
 70 DATA 1010,1150,1240,1310,
 1410,1460,1500,1360,1080,1550
 ,3110
300 REM **ACHAR O LOCAL**
310 CLS
330 IF L<11 THEN  GOSUB G(L,1)
: GOTO 400
340 IF L<21 THEN  GOSUB G(L-10
,2): GOTO 400
350 IF L<26 THEN  GOSUB G(L-20
,3)
```

```
300 REM **ACHAR LOCAL**
310 CLS
330 IF L<11 THEN ON L GOSUB 0,0
,0,5020,0,0,5050,5080,0,5110:GO
TO 400
340 IF L<21 THEN ON L-10 GOSUB
5140,0,0,5180,5210,5240,5270,53
00,0,0:GOTO 400
350 IF L<26 THEN ON L-20 GOSUB
0,5330,0,5360
```

Antes de escrever esse tipo de rotina, verifique se o número correspondente a cada descrição de localização está correto. Começando com a localização 1, faça uma lista dos números de linhas correspondentes a cada descrição. Se não houver descrição para uma determinada localização, escreva o número 0. Nesta aventura, já dispomos de descrição para a localização de número 4, mas não para as de número 1, 2 e 3.

De posse do conjunto contendo os números de linhas correspondentes à descrição de cada localização, você pode começar a escrever a rotina. Em todos os computadores, exceto o Spectrum, esta rotina é constituída de uma seqüência de operações que verifica o valor da variável **L** e usa o comando **ON...GOSUB**. No caso do Spectrum, que não possui este comando, o número das linhas é obtido de uma matriz.

Nos outros programas, o conjunto contendo os números de linhas está nas instruções colocadas nas linhas 330 a 350. Inicia com a localização 1 no começo da linha 330 e termina com a localização 24 no final da linha 350.

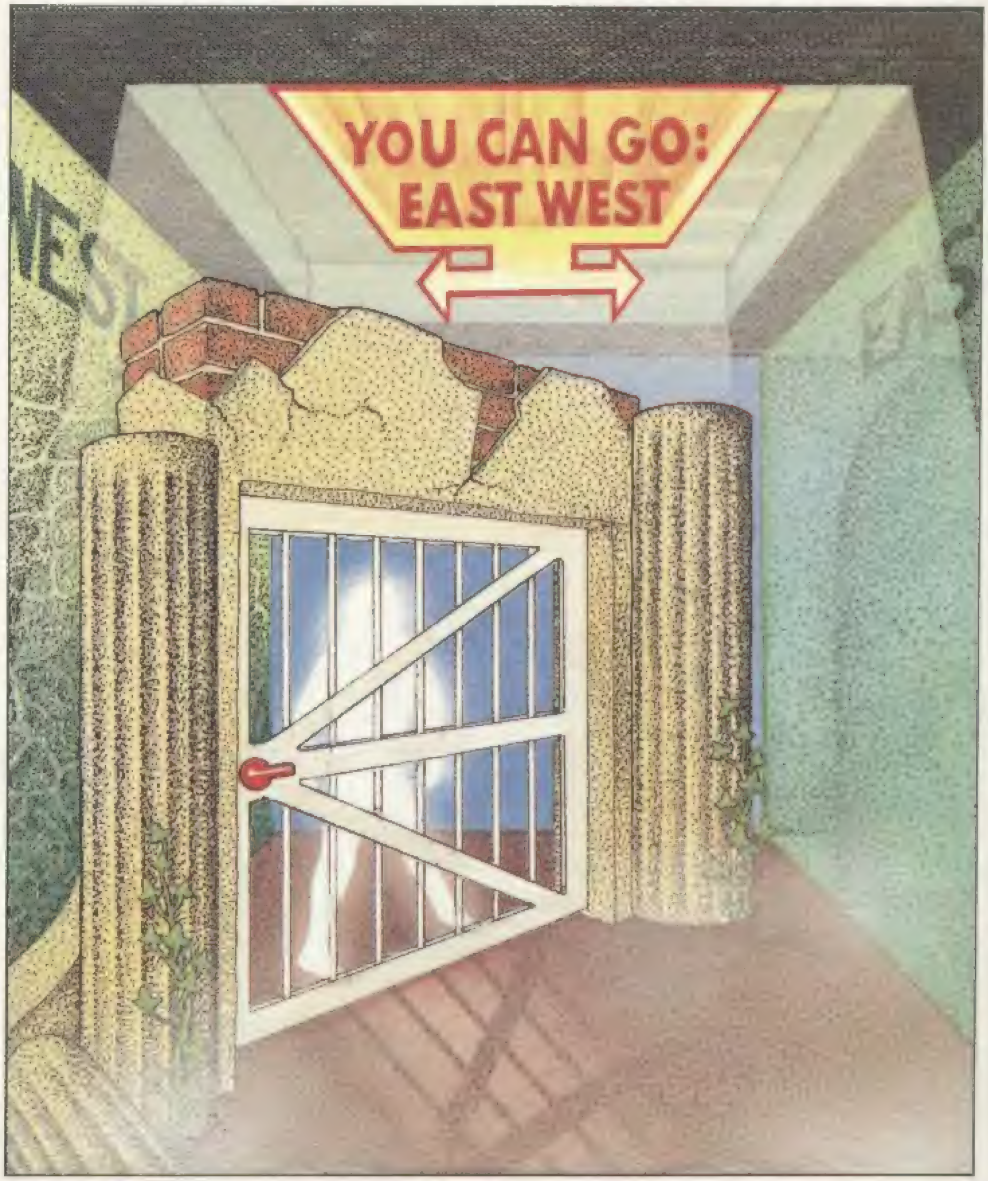
No Spectrum, use o valor de **L** para ter acesso a um elemento da matriz construído pelas instruções apresentadas nas linhas 20 e 30. Note que a matriz está superdimensionada em relação à quantidade de locais existentes na aventura. Os números excedentes são todos iguais a zero e não têm nenhuma influência no desenrolar da aventura. O dimensionamento extra foi feito porque a matriz será utilizada em partes do programa que mais tarde apresentaremos.

Uma observação válida somente para os usuários do Spectrum: o comando **POKE**, na linha 20, faz com que o computador aceite apenas letras maiúsculas. Isto evitará problemas no momento de se comparar as respostas dadas pelo jogador com as armazenadas na variável **R$**.

APRESENTAÇÃO DAS DIREÇÕES

Além da descrição do local onde se encontra, o jogador também precisa conhecer as saídas disponíveis. O programa deve verificar, pelas variáveis **N**, **E**, **S**, e **W**, quais são estas saídas. A informação é necessária porque nem sempre se pode sair em todas as direções, estando em um determinado local. A próxima seção do programa indica as saídas.

```
390 REM **APONTAR DIRECOES**
```



```
400 IF DA<>1 THEN  PRINT '"POD
E SEGUIR";
410 IF N>0 THEN  PRINT TAB 11;
"NORTE"
420 IF E>0 THEN  PRINT TAB 11;
"LESTE"
430 IF S>0 THEN  PRINT TAB 11;
"SUL"
440 IF W>0 THEN  PRINT TAB 11;
"OESTE"
```

```
390 REM **INDICAR DIRECOES**
400 IF L<>11 OR (LA=1 AND OB(6)
=-1) THEN PRINT:PRINT"PODE SEGU
IR ";:ELSE 460
410 IF N>0 THEN PRINT TAB(13);"
NORTE"
420 IF E>0 THEN PRINT TAB(13);"
LESTE"
430 IF S>0 THEN PRINT TAB(13);"
SUL"
440 IF W>0 THEN PRINT TAB(13);"
OESTE"
```

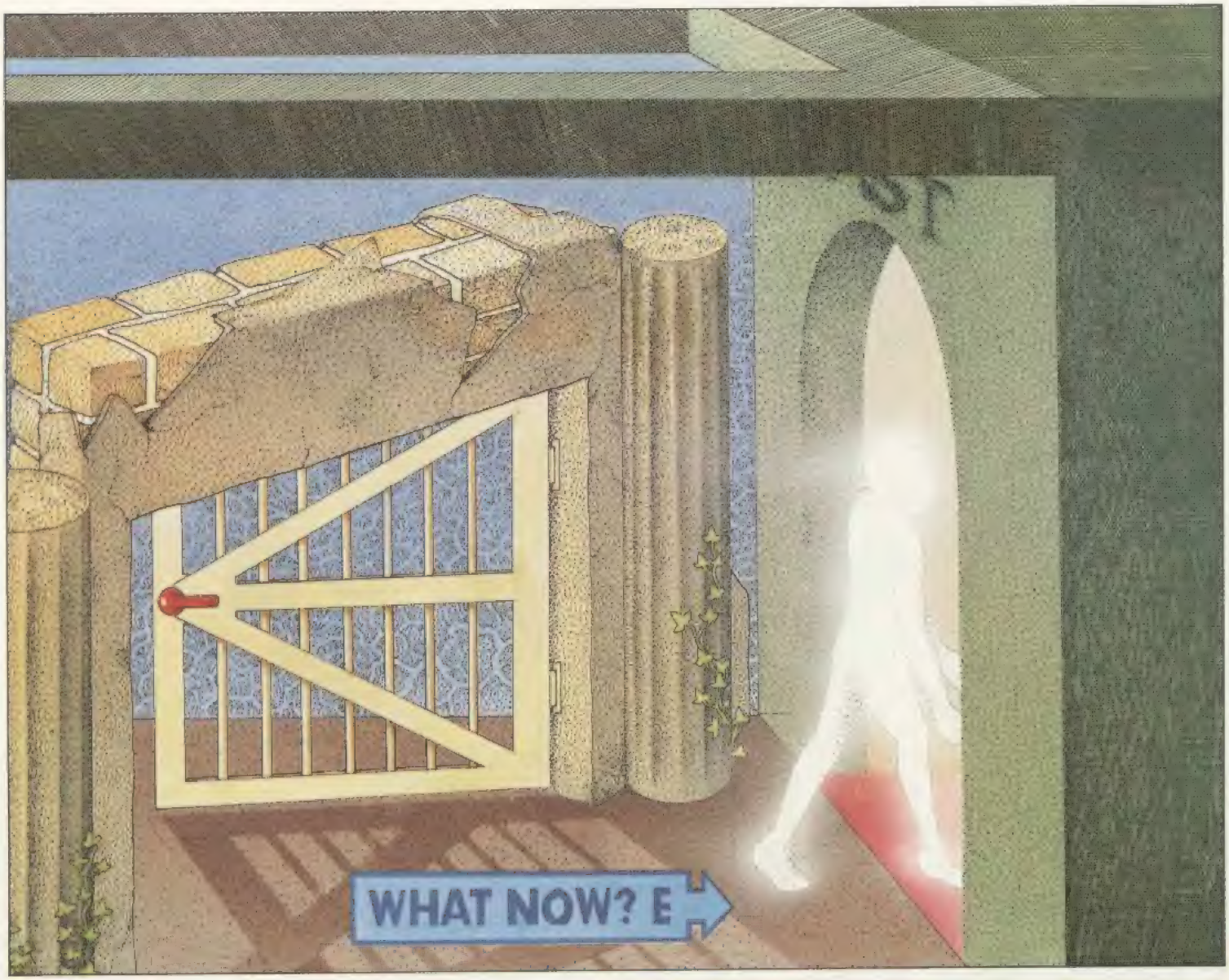
```
390  REM  **APONTAR DIRECOES**
400  IF L <  > 11 OR (LA = 1 AN
D OB(6) =  - 1 THEN  PRINT : PR
INT "PODE SEGUIR ";: GOTO 410
405  GOTO 460
410  IF N > 0 THEN  PRINT  TAB(
 11);"NORTE"
420  IF E > 0 THEN  PRINT  TAB(
 11);"LESTE"
430  IF S > 0 THEN  PRINT  TAB(
 11);"SUL"
440  IF W > 0 THEN  PRINT  TAB(
 11);"OESTE"
```

A rotina apresentada simplesmente verifica os valores das variáveis **N**, **S**, **E**, e **W** que você definiu, baseado em seu mapa de localidades. Se o valor da variável for maior do que zero, a direção escolhida é uma direção possível — ou seja, é uma saída. Essa rotina pode ser utilizada, sem modificações, em qualquer aventura que empregue um setoramento desse tipo.

INSTRUÇÕES

Agora que o jogador conhece as direções possíveis, podemos perguntar-lhe: "**PARA ONDE?**". A rotina responsável por esta pergunta é apresentada abaixo.

```
450 REM **INSTRUCOES**
460 INPUT  INVERSE 1;"PARA OND
E "; LINE I$
470 GOSUB 3010
515 GOTO G(I,4)
```



```
450 REM **INSTRUCOES**
460 PRINT:INPUT"PARA ONDE";I$
470 GOSUB 3010
```

A resposta dada pelo jogador é armazenada na variável **I$**. O programa verificará o tipo de resposta e atuará de acordo.

A próxima linha — linha 470 — desvia o fluxo do programa para uma sub-rotina localizada na linha 3010. Esta sub-rotina é responsável pela validação da resposta do jogador.

```
 600 REM **ROTINA INSTR.**
 610 LET IN=0: IF LEN Y$>LEN X$
  THEN  RETURN
 620 FOR K=1 TO (LEN X$-LEN Y$+
 1)
 630 IF Y$=X$(K TO K+LEN Y$-1)
 THEN  LET IN=K: GOTO 650
 640 NEXT K
 650 RETURN
3000 REM **ROTINA VERIFICACAO**
3010 LET N$="": LET X$=I$: LET
Y$=" ": GOSUB 600: LET I=IN
3020 IF I=0 THEN  LET V$=I$: GO
TO 3050
3030 LET V$=I$( TO I-1)
3040 LET N$=I$(I+1 TO )
3050 LET I=0
3060 FOR K=1 TO 19
3070 IF V$=R$(K, TO LEN V$) THE
N  LET I=R(K): LET I$=V$( TO 1)
3080 NEXT K
3090 RETURN
```

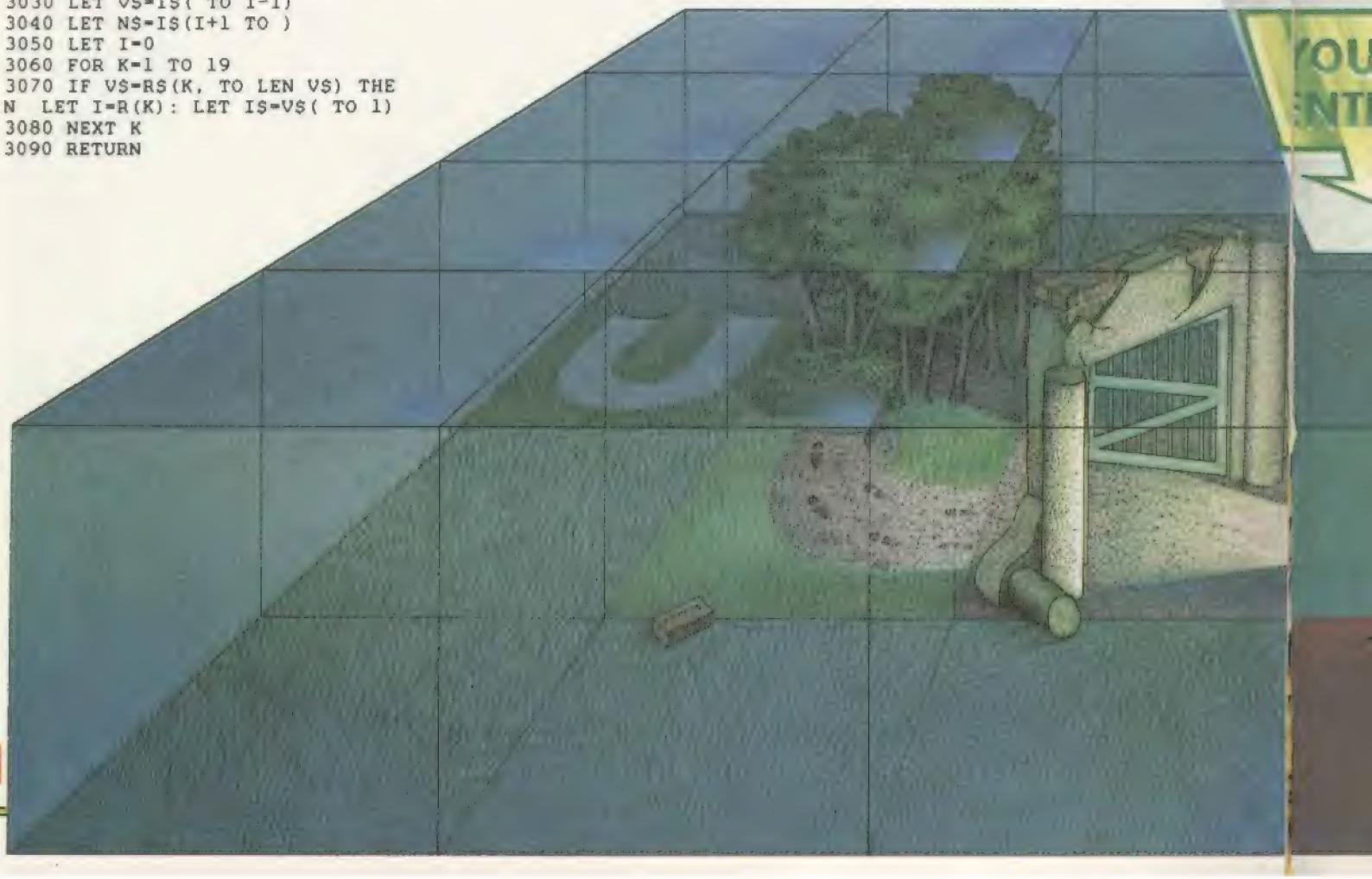
```
3000 REM **INSTRUCAO DE CHECAGE
M**
3010 N$="":I=INSTR(I$," ")
3020 IF I=0 THEN V$=I$:GOTO 305
0
3030 V$=LEFT$(I$,I-1)
3040 N$=MID$(I$,I+1)
3050 I=0
3060 FOR K=1 TO 19
3070 IF INSTR(R$(K),V$)=1 THEN
I=R(K):I$=LEFT$(V$,1)
3080 NEXT
3090 RETURN
```

```
3000  REM  **ROTINA DE VERIFICA
CAO**
3010 N$ = "": FOR Z = 1 TO  LEN
 (I$): IF  MID$ (I$,Z,1) = " "
THEN I = Z: GOTO 3020
3015  NEXT :I = 0
3020  IF I = 0 THEN V$ = I$: GO
TO 3050
3030 V$ =  LEFT$ (I$,I - 1)
3040 N$ =  MID$ (I$,I + 1)
3050 I = 0
3060  FOR K = 1 TO 19
3070  IF V$ =  LEFT$ (R$(K), LE
N (V$)) THEN I = R(K):I$ =  LEF
T$ (V$,1)
3080  NEXT
3090  RETURN
```

A rotina verifica se a variável **I$** compõe-se de duas palavras. Em caso afirmativo, a primeira palavra é chamada de **V$** e a segunda de **N$**. A variável **V$** contém um verbo — como **PEGAR, MATAR, LEVAR** — e todas as palavras indicativas de direção: **NORTE, SUL, LESTE** e **OESTE**. A variável **N$** armazena os nomes dos objetos que fazem parte da aventura.

Os computadores TRS-Color e MSX usam o comando **INSTR**, na linha 3010, para verificar se há algum espaço na resposta que foi armazenada em **I$**. Este espaço será indicativo da existência de duas palavras: a que pertence a **V$** e a que pertence a **N$**. No Spectrum e no Apple não existe o comando **INSTR**. Para substituí-lo, usamos uma pequena sub-rotina, colocada nas linhas 600 a 650.

Se um espaço for encontrado, a instrução na linha 3030 separa **I$** em suas componentes **V$** e **N$**. Se não houver espaço, a linha 3020 considera **V$** igual a **I$**.



A parte final da sub-rotina, composta pelas linhas 3060 a 3080, compara as respostas dadas com o conteúdo da matriz **R$**. Como sabemos, a matriz **R$** contém as palavras indicativas das direções que podem ser seguidas. Depois você verá como fazer para expandir o conteúdo da matriz **R$**. Pela instrução da linha 3070, o programa verifica se há igualdade entre os conteúdos **R$** e **V$**. Se houver, a variável **I** assume o valor da variável **R**. O programa reconhecerá a igualdade dos conteúdos verificando se o valor de **I** é maior que zero. A última parte da linha 3070 retira a primeira letra de **V$** e armazena-a na variável **I$**. A variável **I$** será utilizada depois, para fazer com que o jogador se movimente.

As sub-rotinas apresentadas adaptam-se a qualquer aventura, sem necessidade de grandes modificações. Apenas um detalhe pode precisar de alguns ajustes: a duração do comando **FOR...NEXT**, na linha 3060.

MOVIMENTOS

O passo seguinte consiste em adicionar uma rotina destinada a manipular a variável **L**, indicativa de localização, de acordo com o valor assumido pela variável **I$**. Esta rotina é apresentada em seguida:

```
1000 REM **ROTINA MOVIMENTO**
1010 IF I$="N" AND N>0 THEN  LE
T L=L-6: GOTO 310
1020 IF I$="L" AND E>0 THEN  LE
T L=L+1: GOTO 310
1030 IF I$="S" AND S>0 THEN  LE
T L=L+6: GOTO 310
1040 IF I$="O" AND W>0 THEN  LE
T L=L-1: GOTO 310
1050 REM **SE NAO HA LOCAL POSS
IVEL NESSA DIRECAO**
1060 PRINT '"DESCULPE, VOCE NAO
 PODE SEGUIR  NESTA DIRECAO.":
GOTO 330
```

```
1000 REM **ROTINA DE MOVIMENTO*
*
1010 IF I$="N" AND N>0 THEN L=L
-6:GOTO 310
1020 IF I$="L" AND E>0 THEN L=L
+1:GOTO 310
1030 IF I$="S" AND S>0 THEN L=L
+6:GOTO 310
1040 IF I$="O" AND W>0 THEN L=L
-1:GOTO 310
1050 REM **SE NAO HOUVER LOCAL
POSSIVEL NESSA DIRECAO**
1060 PRINT:PRINT"DESCULPE - VOC
E NAO PODE SEGUIR POR ESTE CAMI
NHO.":GOTO 330
```

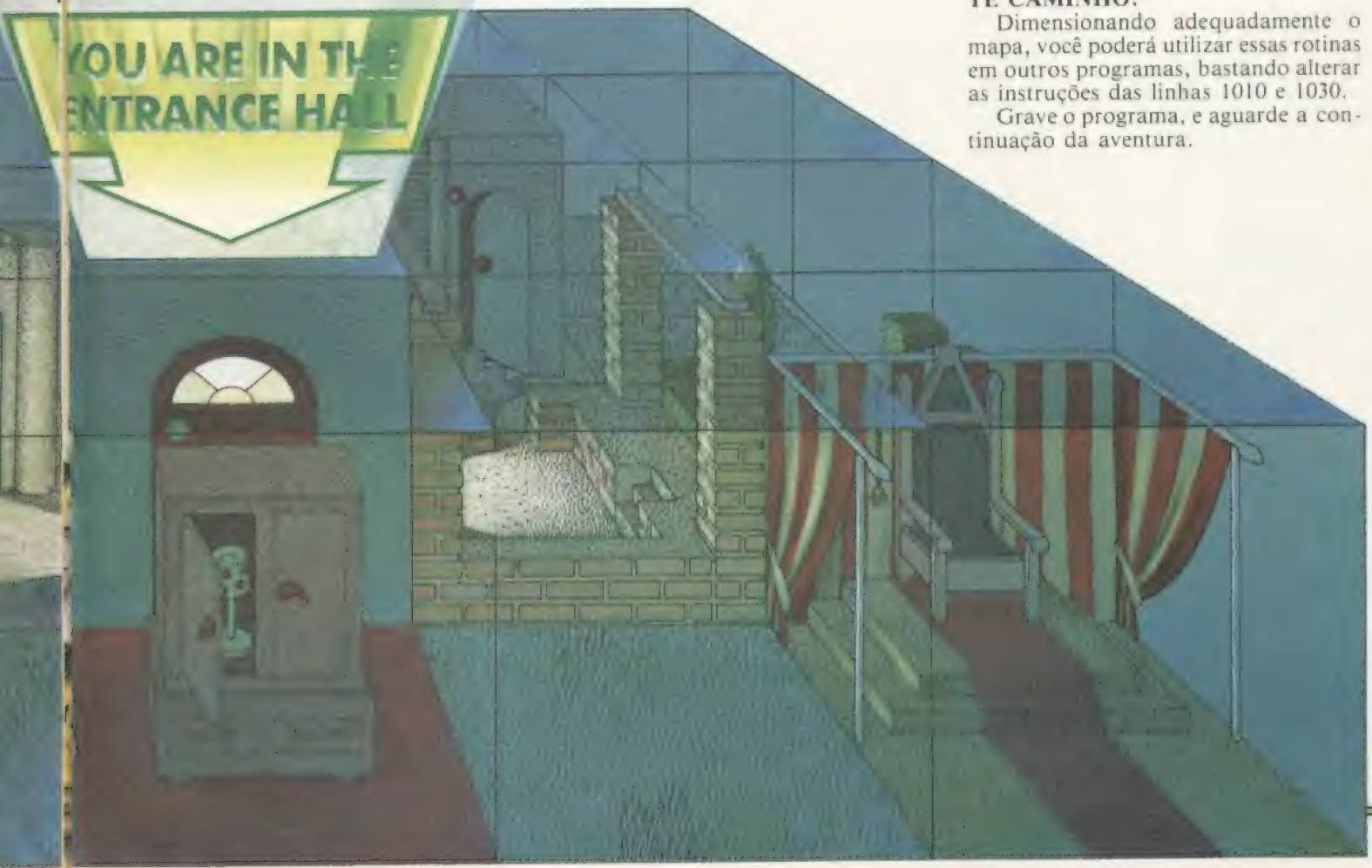
Como você está lembrado, o ponto de partida da aventura foi um mapa com uma largura de seis posições. Mover o jogador por essas posições significa alterar o valor da variável **L** por um fator baseado no tamanho do mapa. Por exemplo, para fazer com que o jogador se movimente nas direções Norte ou Sul basta adicionar ou subtrair seis da variável **L**. Isso fará com que o jogador suba ou desça uma linha completa no mapa. De modo semelhante, mover o jogador nas direções Leste ou Oeste significa adicionar ou subtrair 1 de **L**.

As instruções das linhas 1010 a 1040 verificam o conteúdo da variável **I$** e ajustam o valor de **L**. As saídas possíveis são definidas nas linhas que seguem as descrições dos locais.

Se não há uma saída na direção que o jogador escolheu, a linha 1060 apresenta a mensagem: "**DESCULPE — VOCÊ NÃO PODE SEGUIR POR ESTE CAMINHO.**"

Dimensionando adequadamente o mapa, você poderá utilizar essas rotinas em outros programas, bastando alterar as instruções das linhas 1010 e 1030.

Grave o programa, e aguarde a continuação da aventura.



DATILOGRAFIA: ALFABETO COMPLETO

Se você já dominou o uso das teclas centrais, é hora de treinar a datilografia com todo o alfabeto. Para isso, veja como alterar o programa apresentado no artigo anterior.

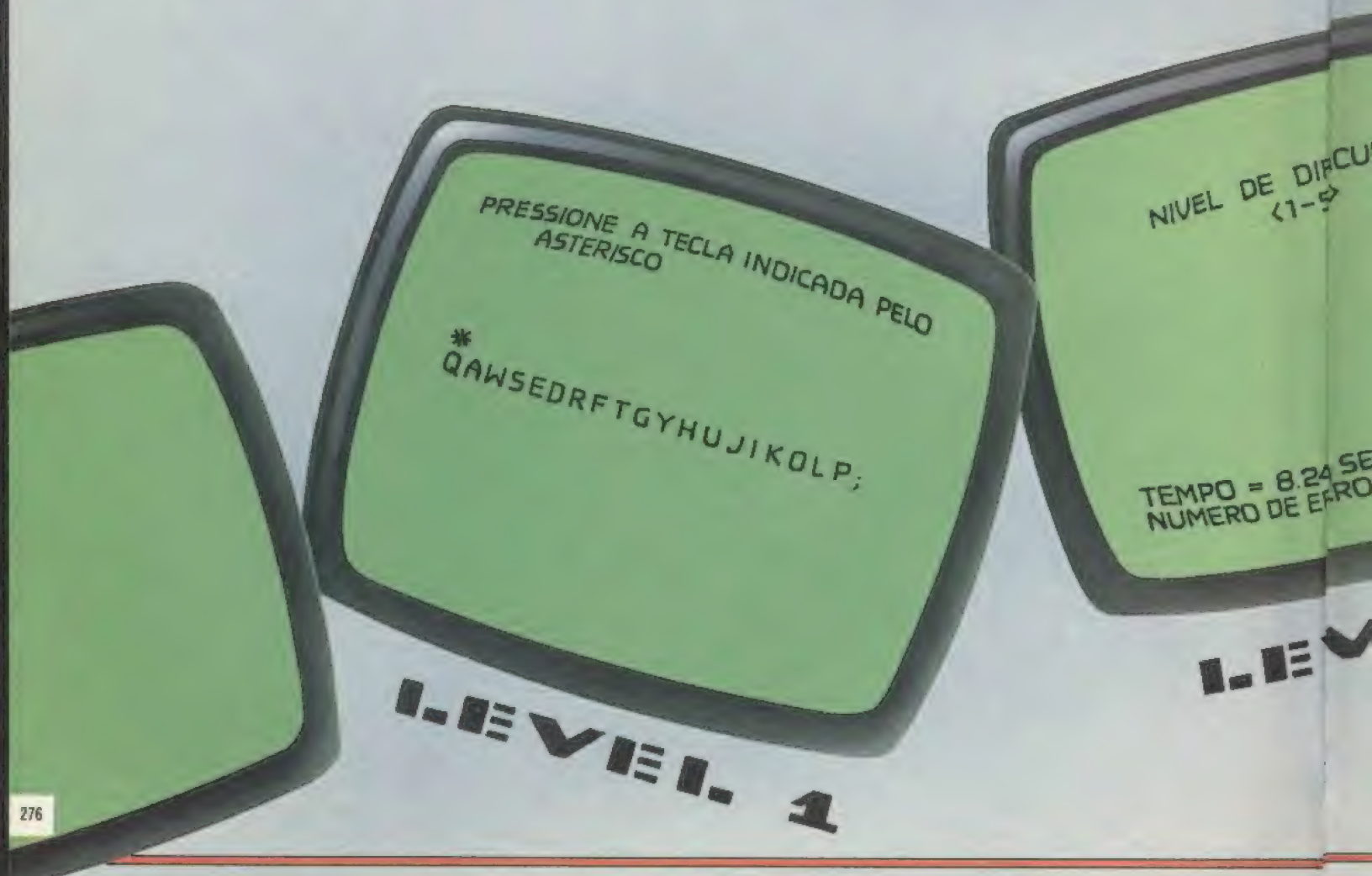
No artigo da página 249 apresentamos a primeira parte de um programa completo para aprender datilografia. Tendo já dominado razoavelmente o uso das teclas da fileira do meio — ou seja, se você datilografa cerca de 15 palavras por minuto, sem cometer nenhum erro —, é hora de passar para o segundo estágio.

AS TECLAS QWERTY

Atrescentando as linhas que se seguem ao programa do artigo anterior, você poderá treinar também as teclas da fileira de cima do teclado: a chamada fileira **QWERTY** (é desta fileira que vem o nome dado aos teclados usados em computadores).

Carregue o programa antigo e digite as linhas que se seguem conforme o tipo de seu microcomputador. Algumas delas substituirão linhas previamente existentes — que agora se tornaram desnecessárias —, ao passo que outras são totalmente novas.

```
 30 LET S$="QAWSEDRFTGYHUJIKOL
 p"
210 FOR K=6 TO 24
230 LET R$=S$(K-5)
 320 LET RN=INT (RND*19)+1
 330 PRINT AT 10,RN+5,;"*": LET
R$=S$(RN)
 350 PRINT AT 10,RN+5;" "
 440 LET RN=INT (RND*19)+1
 530 PRINT AT 10,13;"
": PRINT AT 10,13;T$
 540 FOR M=1 TO LEN T$: PRINT
 AT 9,11+M;" *          "
 610 FOR N=1 TO 4: RESTORE :
 LET RN=INT (RND*24)+1: FOR K=1
 TO RN: READ X$: NEXT K
1010 PRINT AT 12,6;S$
2000 DATA "QUEDA","TIRO","TROLE
","POLE","GRALHA","RISO","PORTA
","PATRULHA","URSO","PILHA","RU
A","ILHA"
2010 DATA "TULIPA","PIOLHO","IO.
```



- COMO ACRESCENTAR AS TECLAS DAS FILEIRAS SUPERIOR E INFERIOR AO PROGRAMA
- POSIÇÃO DOS DEDOS NAS TECLAS DE OUTRAS FILEIRAS
- COMO MUDAR AS PALAVRAS PARA TREINAMENTO
- PRATIQUE COM TODO O ALFABETO

```
DO","PIADA","JAULA","FLORESTA",
"OLHO","PODER","RATO","TROPA","
PISTOLA","QUADRA"
```

```
10 OB$="QAWSEDRFTGYHUJIKOLP"
210 AP=1253
220 FOR K=1 TO 19
230 AP=AP+1
320 AP=1253+RND(19)
430 P$=MID$(OB$,RND(19),1)
800 CLS:P$="":FOR K=1 TO 4
9000 DATA QUARTO,LADO,FATOS,SER
IA,GALHO,JATO,HARPA,LOTERIA,POR
TA,QUILHA,SADIO,RALA,ORLA,ATILA
9010 DATA EDITAR,TALHER,DIRETO,
ULISSES,ILHA,FOGO,JULGAR,ASSADO
,DERROTA,HULHA,GASODUTO,LATA,TI
JOLO
```

```
10 OB$="QAWSEDRFTGYHUJIKOLPÇ":Z
$=CHR$(219):S$="L10 O2 G"
210 AP=358
```

```
10 OB$ = "Q A W S E D R F T G Y
 H U J I K O L P ;"
210 AP =  - 1
220  FOR K = 1 TO 20:AP = AP +
2: GOSUB 1100: NEXT
320 AP = 1 +  INT ( RND (1) * 2
0) * 2
420  FOR K = 1 TO 20:P$ =  MID$
 (OB$, INT ( RND (1) * 20) * 2
+ 1,1)
610  FOR K = 1 TO 5:P$ = P$ + W
$( INT ( RND (1) * 28) + 1) +
```

```
220 FOR K=1 TO 20
230 AP=AP+1
320 AP=359+INT(RND(1)*20)
430 P$=MID$(OB$,INT(RND(1)*20)+
1,1)
810 FOR K=1 TO 4:P$=P$+W$(INT(R
ND(1)*28)+1)+CHR$(32):NEXT
9000 DATA QUARTO,LADO,FATOS,SER
IA,GALHO,JATO,HARPA,LOTERIA,POR
TA,QUILHA,SADIO,RAÇA,ORLA,ATIÇA
R
9010 DATA EDITAR,TALHER,DIRETO,
ULISSES,ILHA,FOGO,JULGAR,JULHO,
ASSADO,DERROTA,HULHA,GASODUTO,L
AÇO,TIJOLO
```

```
CHR$ (32): NEXT
1010  VTAB 12: HTAB 1: PRINT OB
$
9000  DATA  QUARTO,LADO,FATOS,
SERIA,GALHO,JATO,HARPA,LOTERIA,
PORTA,QUILHA,SADIO,RALA,ORLA,AT
ILA
9010  DATA  EDITAR,TALHER,DIRE
TO,ULISSES,ILHA,FOGO,JULGAR,JUL
HO,ASSADO,DERROTA,HULHA,GASODUT
O,LACO,TIJOLO
```

Quando o programa for executado, a tela exibirá o menu usual, com cinco opções. O nível 1 exibe a seqüência de letras **QAWSEDRFTGYHUJIKOLP**. Dependendo de seu computador, o teclado também incluirá um caractere final: ; ou Ç. Da mesma forma que o programa anterior, um asterisco imediata-

mente abaixo das letras na tela funcionará como um sinal de prontidão, caminhando da esquerda para a direita.

Os níveis 2 e 3 funcionam exatamente como o primeiro, levando você a digitar os caracteres aleatoriamente, mas com uma série muito maior de letras.

Os níveis 4 e 5 são mais difíceis do que os anteriores, porque envolvem a digitação de palavras mais extensas, formadas por letras das duas diferentes fileiras.

Para começar a usar o programa, sente-se diante do teclado e posicione os dedos corretamente na fileira do meio. Tente digitar as teclas da linha superior movimentando apenas o dedo que for necessário para cada uma e não toda a mão (veja a ilustração). Utilize o dedo mínimo da mão esquerda para acionar o **Q** ou o **A**, o dedo anular para o **S** e o **W**, e assim por diante, até o dedo mínimo da mão direita, que deve ser usado para acionar a tecla **P**. Os dedos indicadores trabalharão mais que todos — o indicador esquerdo será utilizado para o **F**, **G**, **R** e **T**, enquanto o indicador direito se encarregará das letras **H**, **J**, **Y** e **U**. Depois de ter pressionado uma tecla da fileira superior, volte com o dedo à posição inicial, na tecla de apoio.

AS TECLAS ZXCV

Quando estiver digitando com precisão e rapidez as teclas da fileira superior juntamente com as da fileira central, comece a usar as teclas da fileira inferior (Z, X, C, V, B, etc.)

Nesta parte do curso, você ainda não trabalhará com as três fileiras. Antes disso, aprenderá a utilizar as teclas das fileiras central e inferior do teclado, simultaneamente, com exercícios em cinco níveis de dificuldade.

Eis aqui as alterações que deverá efetuar:

S

```
 30 LET S$="AZSXDCFVGBHNJMKL"
210 FOR K=6 TO 21
320 LET RN=INT (RND*16)+1
440 LET RN=INT (RND*16)+1
610 FOR N=1 TO 5: RESTORE :
 LET RN=INT (RND*24)+1: FOR K=1
 TO RN: READ X$: NEXT K
1010 PRINT AT 12,6;"S$"
2000 DATA "CASA","FACA","VAZA",
"LAVA","MACA","VACA","BABACA","
JACA","JAZZ","BANDA","BALA","CA
LCA"
2010 DATA "SALVA","CALA","AMA",
"FALA","DA","AFAGA","ALCANCA","
MANHA","CANA","LAZANHA","SAGA",
"CHAMA"
```

T

```
10 OB$="AZSXDCFVGBHNJMKL"
220 FOR K=1 TO 16
320 AP=1253+RND(16)
430 P$=MID$(OB$,RND(16),1)
800 CLS:P$="":FOR K=1 TO 5
9000 DATA FACA,LAVA,LAMA,BALANC
O,VAGA,NASA,JACA,SAMBA,MACA,AVA
NCA,VANDA,AJAX,CHAMADA,VALHALA,
CALHA,CANSADA
9010 DATA LAMBADA,BANANA,GAMBA,
BANDA,CANA,ABAFA,CANALHA,MASSA,
ZAGA,MANHA,CASCA,SALVA
```

X

```
10 OB$="AZSXDCFVGBHNJMK,L.Ç":Z$
=CHR$(219):S$="L10 O2 G"
220 FOR K=1 TO 19
320 AP=359+INT(RND(1)*19)
430 P$=MID$(OB$,INT(RND(1)*19)+
1,1)
9000 DATA FACA,LAVA,LAMAÇAL,BAL
ANÇA,VAGA,NASA,JACA,SAMBA,MACA,
AVANÇA,VANDA,AJAX,CHAMADA,VALHA
LA,CALHA,CANSADA
9010 DATA LAMBADA,BANANA,GAMBA,
BANDA,CANA,ABAFA,CANALHA,MASSA,
ZAGA,MANHA,CAÇA,BAÇA
```

Ó

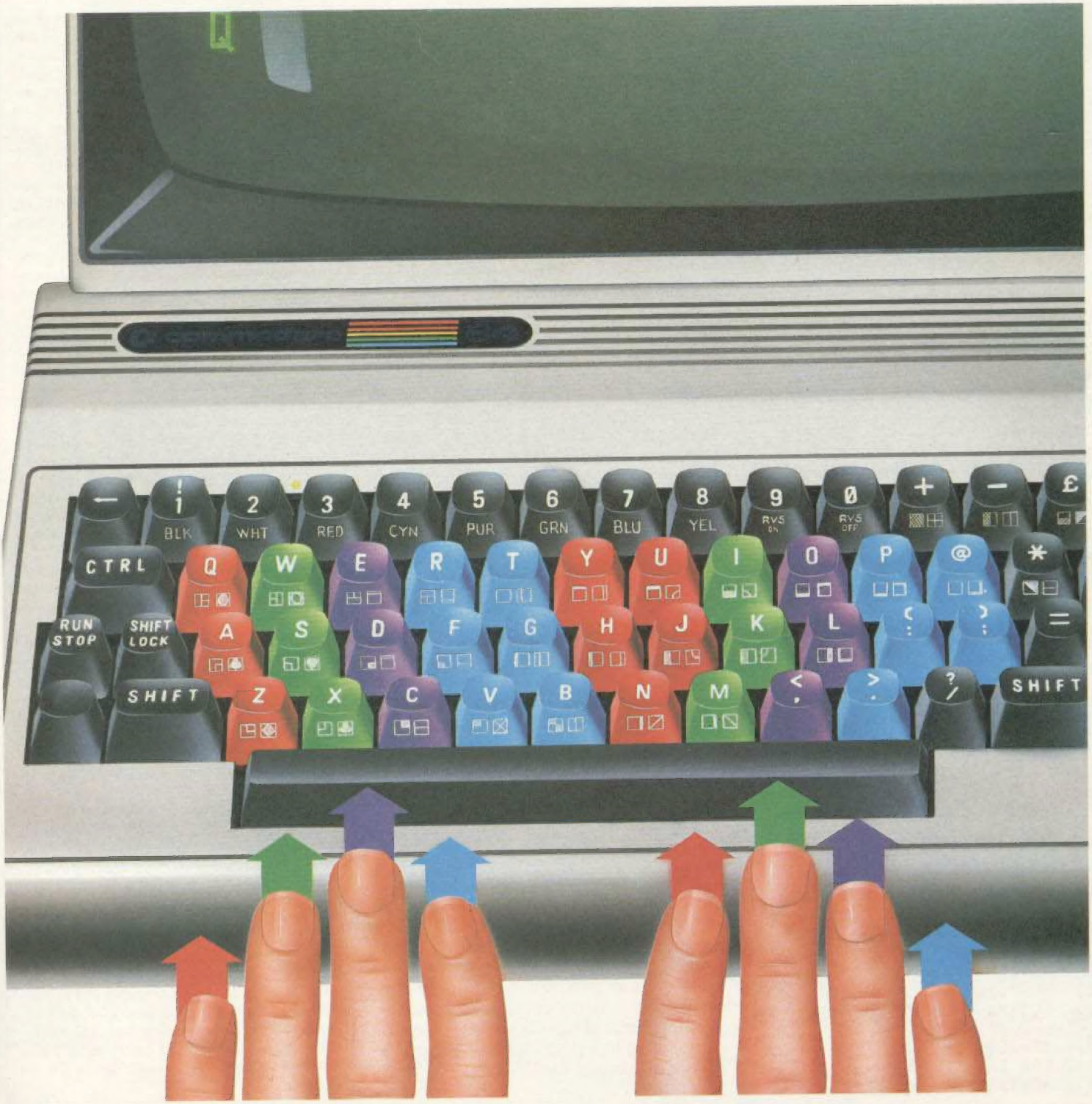
```
10 OB$ = "A Z S X D C F V G B H
 N J M K , L . ;"
220  FOR K = 1 TO 19:AP = AP +
2: GOSUB 1100: NEXT
320 AP = 1 +  INT ( RND (1) * 1
9) * 2
420  FOR K = 1 TO 20:P$ =  MID$
 (OB$, INT ( RND (1) * 19) * 2
+ 1,1)
9000  DATA  FACA,LAVA,LAMA,BALA
NCA,VAGA,NASA,JACA,SAMBA,MACA,A
```

```
VANCA,VANDA,AJAX,CHAMADA,VALHAL
A,CALHA,CANSADA
9010 DATA LAMBADA,BANANA,BAN
DA,CANA,ABAFA,CANALHA,MASSA,ZAG
A,MANHA,CALA,BALA,GAMBA
```

Com esse programa, você também fará exercícios em cinco níveis de dificuldade, mas com palavras que utilizam as letras das fileiras central e inferior do teclado.

Mais uma vez, sente-se diante do teclado, com os dedos posicionados corretamente. Agora você movimentará seus dedos para baixo, para a fileira inferior, antes de retornar às teclas de apoio. O dedo mínimo da mão esquerda irá pressionar o **A** e o **Z**, o anular o **S** e o **X** e assim por diante. O dedo indicador da mão esquerda será utilizado para as letras **F**, **G**, **V** e **B**, e o indicador da mão direita para **H**, **J**, **N** e **M**. Os dedos restantes deverão operar as teclas de pontuação situadas na fileira inferior, em posições que variam de teclado para teclado. No Spectrum, elas não estão disponíveis sem o **<SYMBOL SHIFT>**, cujo emprego será explicado futuramente.

Observe que essas teclas não estão incluídas nos testes de palavras — você deverá praticar a pontuação com um programa adicional, que apresentaremos em artigo posterior.



O ALFABETO COMPLETO

Depois de dominar o uso conjunto das teclas das fileiras superior e inferior do teclado, você estará apto a ingressar no estágio seguinte do curso. Afinal, você poderá praticar a datilografia utilizando todo o alfabeto. Apenas as teclas de números e de pontuação permanecerão temporariamente excluídas do treino.

Eis aqui as alterações que deverá efetuar no programa:

```
 30 LET S$="QAZWSXEDCRFVTGBYHN
 UJMIKOLP"
 40 FOR K=2 TO 27
230 LET R$=S$(K-1)
320 LET RN=INT (RND*26)+1
330 PRINT AT 10,RN+1;"*": LET
R$=S$(RN)
350 PRINT AT 10,RN+1;" "
440 LET RN=INT (RND*26)+1
530 PRINT AT 10,13;"
": PRINT AT 10,13;T$
540 FOR M=1 TO LEN T$: PRINT
AT 9,11+M;" *          "
610 FOR N=1 TO 5: RESTORE :
LET RN=INT (RND*24)+1: FOR K=1
TO RN: READ X$: NEXT K
1010 PRINT AT 12,2;S$
2000 DATA "QUIETO","LONGE","ASI
LO","MENTE","LOCAL","TRADICAO",
"RESPOSTA","ATRAVES","DRIBLE","
RETORNO","DESPEDIDA","ESCRITA"
2010 DATA "ESCOLA","INFERNO","P
ROFESSOR","CHATO","TELEVISAO","
BURRA","ESPORTE","BOM","COMPUTA
DOR","MELHOR","INPUT","PERFEITO
"
```

```
10 OB$="QAZWSXEDCRFVTGBYHNUJMIK
OLP"
210 AP=1248
220 FOR K=1 TO 26
320 AP=1248+RND(26)
430 P$=MID$(OB$,RND(26),1)
800 CLS:P$="":FOR K=1 TO 4
1020 PRINT @257,OB$
9000 DATA MARIA,MULHER,JEITO,CH
EGAR,QUENTE,COSTUME,LOCAL,BONIT
A,DECOTE,CONVERSA,UTERO,JAPAO,C
ORRIDA,AZUL
9010 DATA BANDEIRA,AVIAO,VAZIO,
ABERTO,HALITO,XAXADO,ALCALINO,U
NIDOS,PRESSA,QUERIDA,GUARDIAO,A
BOBORA,NAVIO,REMORSO
```

```
10 OB$="QAZWSXEDCRFVTGBYHNUJMIK
,OL.PÇ":Z$=CHR$(219):S$="L10 O2
 G"
210 AP=353
220 FOR K=1 TO 29
320 AP=353+INT(RND(1)*29)
430 P$=MID$(OB$,INT(RND(1)*29)+
1,1)
1010 LOCATE 0,12:PRINTOB$
9000 DATA MARIA,MULHER,JEITO,CH
EGAR,QUENTE,COSTUME,CANÇÃO,BONI
TA,DECOTE,CONVERSA,ÚTERO,JAPÃO,
CORRIDA,AZUL
9010 DATA BANDEIRA,AVIÃO,VAZIO,
ABERTO,HÁLITO,XAXADO,ALCALINO,U
NIDOS,PRESSA,QUERIDA,GUARDIÃO,A
BÓBORA,NAVIO,REMORSO
```

```
10 OB$ = "QAZWSXEDCRFVTGBYHNUJM
IK,OL.P;"
210 AP = 0
220  FOR K = 1 TO 29:AP = AP +
1: GOSUB 1100: NEXT
320 AP = 1 +  INT ( RND (1) * 2
9)
420  FOR K = 1 TO 20:P$ =  MID$
 (OB$, INT ( RND (1) * 29) + 1,
1)
9000  DATA  MARIA,MULHER,JEITO,
CHEGAR,QUENTE,COSTUME,CANCAO,BO
NITA,DECOTE,CONVERSA,UTERO,JAPA
O,CORRIDA,AZUL
9010  DATA  BANDEIRA,AVIAO,VAZI
O,ABERTO,HALITO,XAXADO,ALCALINO
,UNIDOS,PRESSA,QUERIDA,GUARDIAO
,ABOBORA,NAVIO,REMORSO
```

Como acontece com todos os programas deste curso, assim que você estiver familiarizado com as palavras constantes nas declarações **DATA**, poderá trocá-las por novas palavras. Quase sempre, os programas para outras máquinas incluem palavras diferentes. Se quiser, aproveite-as em seu computador, mas verifique se não está utilizando um número de palavras menor que o do programa original. Caso isso ocorra, você obterá uma mensagem de erro **OUT OF DATA**. E, se colocar mais palavras do que as existentes, elas não serão lidas pelo computador, a menos que você modifique o programa.

Siga as cinco lições, como nos estágios anteriores. Lembre-se sempre de colocar os dedos de volta nas posições corretas de apoio, na fileira do meio, toda vez que pressionar uma tecla das carreiras inferior ou superior.

APERFEIÇOAMENTO

Na medida em que o curso for progredindo, você terá a oportunidade de se aperfeiçoar cada vez mais, adquirindo velocidade e precisão na datilografia com todas as letras do alfabeto. Em seguida, começará a praticar com as teclas de números e de pontuação — essenciais para a digitação das listagens de programas.

Como acentuar textos em português em um microcomputador?

Depende muito da linha ou da marca do micro. O problema não foi tecnicamente resolvido de maneira uniforme pelos fabricantes nacionais, pois durante muito tempo não houve um padrão industrial obrigatório. Entre os computadores pessoais cobertos por INPUT, apenas os micros da linha MSX seguem o padrão atual de representação dos sinais característicos da língua portuguesa — o chamado BRASCII, que é a extensão brasileira do código ASCII.

O BRASCII, além de definir os códigos numéricos para as letras acentuadas (á, à, Á, À, ô, etc.), determina ainda a localização padronizada da cedilha e dos acentos grave, agudo, til e circunflexo, no teclado do microcomputador. Portanto, a maneira de usá-los, na datilografia, é idêntica à de uma máquina de escrever. Da mesma forma, o comportamento do teclado e do vídeo, durante a datilografia das letras acentuadas, deve ser igual: por exemplo, ao se pressionar a tecla com o til, este sinal aparece no vídeo, e o cursor fica parado no mesmo lugar; ao se pressionar a letra a, ela aparece imediatamente abaixo do acento.

Os micros de outras linhas (Apple II, TRS-80, TRS-Color e Sinclair), por serem copiados de modelos norte-americanos e ingleses, dão ao problema respostas diferentes. As soluções adotadas variam de um teclado inteiramente compatível com o de uma máquina de escrever (por exemplo, o Microengenho II, da Spectrum, que pertence à linha Apple), até a impossibilidade total de acentuação (micros compatíveis com as linhas Sinclair ZX-81 e TRS-Color).

Alguns micros brasileiros da linha Apple adotaram o padrão do chamado *teclado inteligente*, ou *teclado profissional*, em que certas teclas são usadas para digitar a letra já acentuada, com uma única pressão. A localização destas teclas não coincide, evidentemente, com a de uma máquina de escrever, e seu acionamento depende da pressão simultânea de uma ou mais teclas adicionais de controle. Este procedimento dificulta a datilografia, pois difere muito da maneira natural de se usar uma máquina de escrever e compromete bastante a velocidade da digitação.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

TRS-Color

Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO BASIC

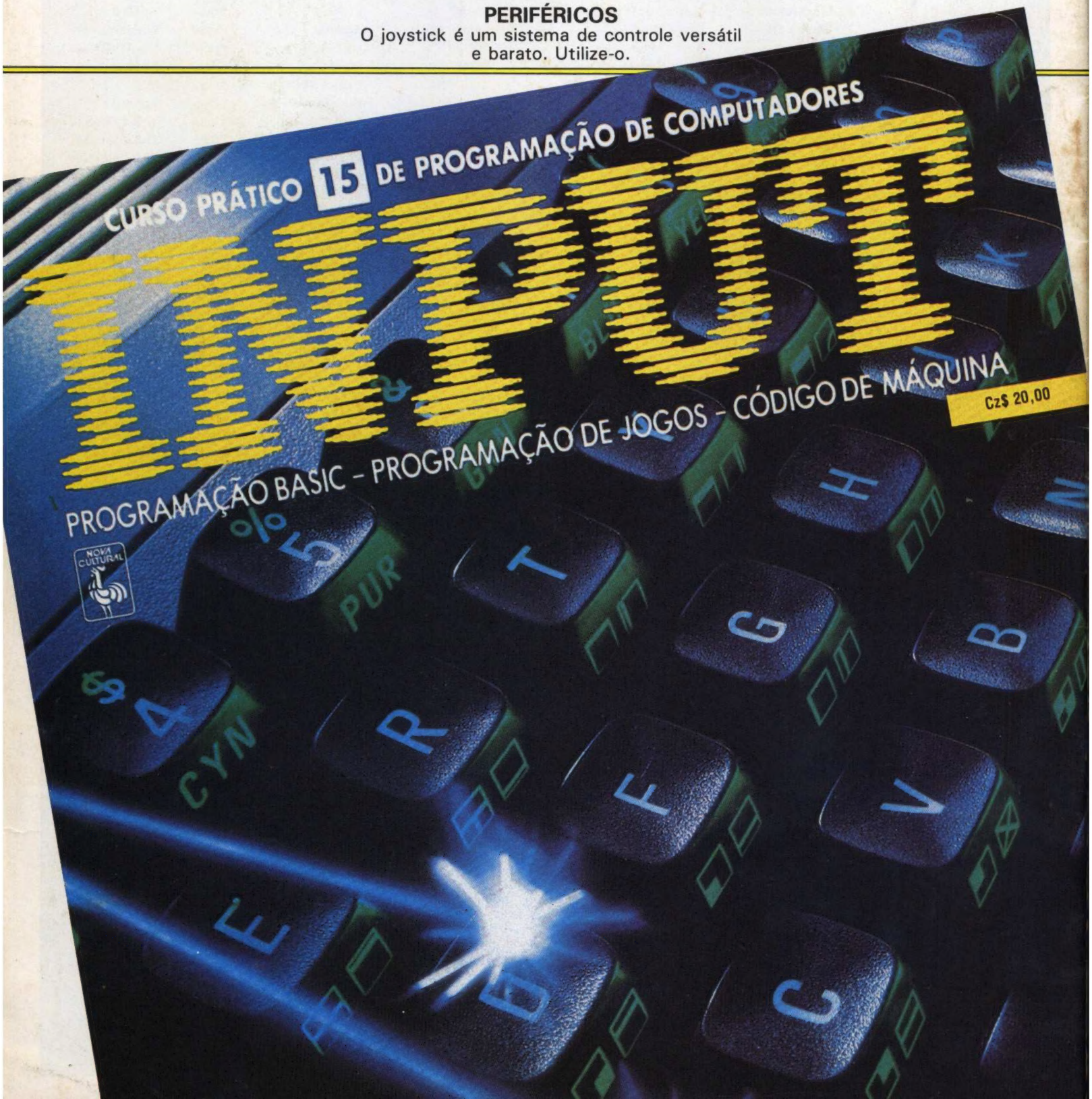
Aperfeiçoe as técnicas de programação estruturada, aprendendo a elaborar uma rotina de ordenação.

APLICAÇÕES

Prosseguindo o curso de datilografia, você verá como incluir no programa os números e símbolos do teclado.

CÓDIGO DE MÁQUINA

Deixe o computador converter o Assembly para código de máquina. Aqui, um Assembler para os usuários do TRS-Color.

PERIFÉRICOS

O joystick é um sistema de controle versátil e barato. Utilize-o.